Fiscal Year 2017 Annual Report

Titled:

Mobility Research for Future Vehicles
A Methodology to Create a Unified Trade-Off Environment for Advanced Aerospace Vehicle

Submitted to the National Institute of Aerospace (NIA) on January 31, 2018

Period of Performance: October 1, 2014 – September 30, 2017

NIA Point of Contact (PoC): Carole E. McPhillips
Deputy Contracts Manager
National Institute of Aerospace
100 Exploration Way
Hampton, VA 23666-6186
(757) 325-6762 (office)
(757) 325-6701 (fax)
carole.mcphillips@nianet.org

ASDL Technical PoC: Dimitri Mavris
Boeing Prof. Advanced Systems Design
dimitri.mavris@aserospace.gatech.edu

Kyle Collins
Research Faculty
kyle.collins@asdl.gatech.edu

Aerospace Systems Design Laboratory
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150
www.asdl.gatech.edu

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
31-01-2018	Annual Research Report	20161001 - 20170930

4. TITLE AND SUBTITLE

Mobility Research for Future Vehicles
A Methodology to Create a Unified Trade-Off Environment for Advanced Aerospace Vehicle

5a. CONTRACT NUMBER

W911NF-16-2-0229

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kyle Collins, Bruce Ahn, Etienne Demers Bouchard, Daniel Bavaro, Ryan Armstrong, Joshua Price, Sylvester Ashok, Matthew Schmit, Dimitri Mavris

5d. PROJECT NUMBER

8502

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Aerospace Systems Design Laboratory, Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0150

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

US Army Research Laboratory
RDRL-VTV
6340 Rodman Rd
Aberdeen Proving Ground, MD 21005

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The development and implementation of advanced aerospace vehicles is an endeavor that can potentially affect long-term aviation operations and future system capabilities for several decades. Selecting the best vehicle configuration(s) requires a thorough understanding of the capabilities and life-cycle considerations required by the end user, the vehicle's full spectrum operations, as well as technologies impacting both operational needs and system performance. The fundamental goal of the proposed effort involves using the Aerospace Systems Design Laboratory (ASDL) established expertise in the fields of decision support and advanced vehicle Modeling and Simulation (M&S) to develop an innovative trade-off environment for advanced vehicle concepts exploration.

15. SUBJECT TERMS

Rotorcraft, tradespace, conceptual design, technology assessment, forecasting, decision making, surrogate model, interactive trade-off, discrete event, use case, maintenance free operating period, rotor optimization, set-based design

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Eric Spero
Unclassified	Unclassified	Unclassified	UU	171	19b. TELEPHONE NUMBER *(Include area code)* (410) 278-8743

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

The development and implementation of advanced aerospace vehicles is an endeavor that can potentially affect long-term aviation operations and future system capabilities for several decades. Selecting the best vehicle configuration(s) requires a thorough understanding of the capabilities and life-cycle considerations required by the end user, the vehicle's full spectrum operations, as well as technologies impacting both operational needs and system performance. The fundamental goal of the proposed effort involves using the Aerospace Systems Design Laboratory (ASDL) established expertise in the fields of decision support and advanced vehicle Modeling and Simulation (M&S) to develop an innovative trade-off environment for advanced vehicle concepts exploration.

Over the span from October 2010 to September of 2017, a Capability Assessment and Trade-off Environment (CATE) with an accompanying Excel user interface was developed. The environment is powered by surrogate models created from the NASA Design and Analysis of Rotorcraft (NDARC) code. The surrogate models were created from data obtained through experiments performed in NDARC using candidate Joint Multi-Role Rotorcraft configurations (Single Main Rotor, Compound, and Tilt-rotor). The use of surrogates for distinct concept families provides a novel way of doing rapid trades to investigate how performance and vehicle unit cost vary across the different designs. To assess technology impacts on vehicle capabilities, CATE includes an Interactive Reconfigurable Matrix of Alternatives (IRMA) that allows for input and management of technologies. CATE uses Quality Function Deployment (QFD) style qualitative analysis for technologies that do not necessarily affect mission performance but do affect mission effectiveness. Users can assess technologies by manually selecting options using the IRMA or by using a genetic algorithm to perform a selection based on the user's objectives. [1]

This fiscal year work aimed to extend the capabilities that currently exist in CATE. To increase the fidelity of the results in CATE, a comprehensive rotor performance analysis using RCAS (Rotor Comprehensive Analysis System) has been used to calibrate a new NDARC model that is then integrated directly into CATE. To increase the accuracy of the calibration, an optimization algorithm has been wrapped around Wayne Johnson's Rotor Performance Spreadsheet, varying the available NDARC variables to best match the calibration data. This process provides a quick and efficient way to calibrate CATE to new models, increasing the tools flexibility and accuracy.

To improve the capabilities of the IRMA in CATE, an extensive rotorcraft technology literature research was performed in order to capture new rotorcraft technologies. During the literature research, different technologies such as ITEM Engine, Continuous Trailing Edge Flap (CTEF) etc., were identified along with their impacts on the various components of the rotorcraft (i.e. physical/functional). These impacts were

then modeled in the CATE environment through the use of tech factors on NDARC parameters. This work ultimately allows for new technologies to be rapidly assessed on a baseline architecture.

In order to extend the actual modeling capabilities, investigation on how OpenMDAO can be used to solve Multidisciplinary Design Analysis and Optimization (MDAO) problems was performed. The open source software was evaluated as a mean to interface with NDARC and perform calculations on the results.

The capabilities of CATE were demonstrated for an existing vehicle, the UH-60 Black Hawk. First, a new procedure to calibrate NDARC files was illustrated for the UH-60A and UH-60L. The power required, power available and component weights were calibrated with published data. Technologies were implemented on the vehicle model and the performance and sizing impacts were derived. Among them, the technologies used to perform the UH-60L to UH-60M upgraded were implemented and the characteristics of the derived UH-60M were analyzed.

A Discrete-Event Simulation (DES) tool was built to model Reliability, Availability, and Maintainability (RAM) of a helicopter performing a mission. This tool can be used to perform system level trade-offs to obtain a desired Operational Availability (Ao), Maintenance Free Operating Period (MFOP), as well as affordability.

2. Framework for characterizing and visualizing the interaction between vehicle performance and effectiveness, and subsystem design parameters

2.1 Introduction

The following sections present the work done in Design Space Exploration and Technology Impact forecasting. The design space exploration works studies the effect of constraints that create a non-hypercubic design space. This work is particularly relevant to rotorcraft because multiple constraints exist in the design space that are either mathematical singularities, incompatible physical variables, incompatible variables that cannot be modeled by software, etc. The technology impact forecasting work presents methods to identify appropriate forecasting methods to use based on the technology under study. This mapping methodology is presented along with a relevant example to the UH-60 helicopter.

This report outlines the steps required to implement the Set-Based Bounded Adaptive Sampling (SeBBAS) approach. The theory of the SeBBAS approach is discussed by Kizer in his PhD Dissertation [2], and is only briefly discussed here to provide sufficient background information to understand the process. The focus of this report is to describe the steps that are required to run the SeBBAS algorithm, which requires both MATLAB and the R programming language. Suggestions are also provided on the most effective way to use this approach.

2.2 Motivation

Common to practically all engineering design problems is the implementation of constraints. Constraints may include some form of design requirement (i.e. wingspan can be no greater than the runway width), performance requirement (i.e. aircraft must have a range exceeding 1000 km), or a physical limitation (i.e. a design that is not physically possible but the mathematical models representing the system still converge). In many of these scenarios, the effects that the constraints have on the design space are not known a-priori, and they often results in a non-hypercubic (NHC) design space.

In addition to constrained design spaces, the engineering models (here an engineering model is taken to be a simplified representation of the true physics of a system) are often complex, non-linear, and expensive to evaluate. In some instances, engineering models are represented as "black-box" functions, in which the user only knows the inputs over some domain, $D = \{x_i \in [a_i, b_i]\}$, and outputs of the model, with no knowledge of what calculations are actually performed within the code. In such instances, it is often advantageous to fit surrogate models to the engineering models, which provide estimates of the engineering models that are inexpensive to evaluate.

Caution must be used when taking a surrogate model approach, as surrogate models generally take the form of continuous functions that are prone to large errors if extrapolation occurs. Extrapolation can normally be avoided by requiring that all design points fall within the domain that was used to create the surrogate models themselves ($x_i \in D$). However, general implementations of surrogate models have no direct way of determining if a design point will be feasible or infeasible. That is, a design point $x_i \in D$ may actually be classified as an infeasible design point by the engineering model, despite being within the domain used to create the surrogate models. Thus, the surrogate model will extrapolate and predict a response for a design point that is actually infeasible due to some constraint or physical limitation within the engineering model. To eliminate this form of extrapolation, an approach capable of classifying a design point as feasible or infeasible (a classification model) must be implemented in conjunction with a surrogate model.

To demonstrate this, consider a 2-Dimensional space defined on the domain $D_1 = \{x_1 \in [0,5], x_2 \in [-5,5]\}$, which is to be evaluated using a black box function that applies some unknown constraint to the design space. The goal is to fit a surrogate model to this constrained black box function. The general procedure for doing so is outlined in Figure 2-1, which requires generating a Design of Experiments (DOE) based on the function domain, and evaluating the black box function at each of these design points to classify them as feasible or infeasible designs. The surrogate model would then be fit to only the feasible design points of the DOE.
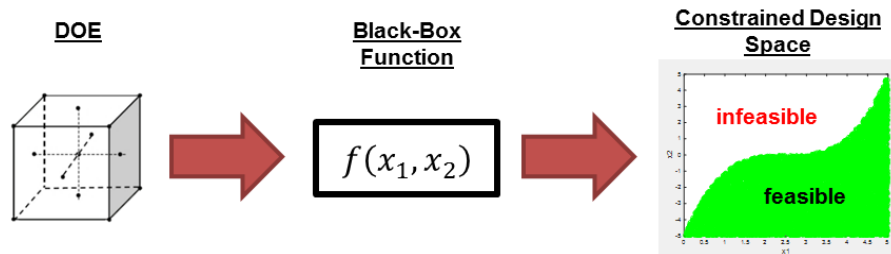
Figure 2-1. Example Test Function that Creates a Non-Hypercubic Design Space Through the Application of a Constraint

Clearly, the constrained design space is NHC (a hypercubic design space would produce feasible designs for all design points within the domain D_1), with a large infeasible design region present within the design space. Also, notice that feasible design points exist for all $x_1 \in [0,5]$ at some constrained subset of x_2 values (i.e. the point $(x_1, x_2) = (2.5, -2.5)$ is feasible, while the point $(2.5, 2.5)$ is infeasible), and likewise feasible points exist for all $x_2 \in [-5,5]$ at some constrained subset of x_1 values. Because of this, a surrogate model fit to the feasible design points will have x values ranging from $x_1 \in [0,5]$ and $x_2 \in [-5,5]$, while only a subset of the domain D_1 is feasible. If the surrogate model is used outside the feasible region of D_1, then the results of the surrogate model are not only complete garbage due to extrapolation,

but are potentially dangerous. That is, it is possible for the surrogate model to extrapolate and predict that a design point has an extremely favorable response, when in reality the design is infeasible. Though this may seem trivial to identify in the simple example provided in Figure 2-1, it becomes exceedingly difficult as the dimensionality of the problem increases beyond three dimensions.

If gone unchecked, the infeasible design point could be carried through parts of the design process, which could potentially cause a lot of costly re-work to be performed at a later stage in the design process. This error can easily be removed by checking the selected design point against the original engineering model, presenting only a minor headache to the designer. A greater challenge is presented if the surrogate model is used in an optimization code to find the optimal design point. If the optimization code has no way of knowing whether a design point is feasible or infeasible, it could potentially always be driven to an infeasible solution, which due to extrapolation, has a favorable response value over other feasible designs. This drives the need for an approach that can identify, with sufficient accuracy, the feasibility of a design point based solely on the values of the design points themselves. This is one of the major objectives of the SeBBAS approach, the implementation of which is described in the following section.

2.3   SeBBAS Approach

2.3.1   Random Forest Algorithm

The SeBBAS approach utilizes an open-source Random Forest (RF) machine learning algorithm (available in the R programming language) to learn and classify any N dimensional design space. The RF algorithm is a type of reinforcement learning algorithm, which uses the provided training data set to construct a set of decision trees. For the SeBBAS algorithm, the decision trees are constructed with the purpose of classifying a given design point as either feasible (given a classification value of 1) or infeasible (given a classification value of -1). In this way, the machine learning algorithm is essentially learning where the boundaries between the feasible and infeasible regions of the design space lie. Once constructed, the RF model has the ability to provide a classification for any design point based on the values of the design variables themselves, as well as predict the probability that any given design point is either feasible or infeasible.

The Random Forest implementation in the SeBBAS algorithm follows the tutorial of Reference [3]. The tutorial provides information on the proper syntax required, as well as details on the available settings of the Random Forest algorithm (such as the number of decision trees used in the model), and how to set the training data for the model. As this part of the SeBBAS algorithm is fully automated and requires no user modifications, no further details are provided in this report. The reader should refer to Brieman provides a more in depth discussion on how to use the Random Forest algorithm in the R programming language. [4]

2.3.2   Incorporation of Random Forest Algorithms into SeBBAS

The Random Forest algorithm is incorporated into SeBBAS as shown in Figure 2-2. First, a lower (α) and upper (β) bound must be established for each of the N design variables being considered. These ranges are then used to construct a DOE to sample the hypercubic design space domain, defined by $D_N = \{x_i \in [\alpha_i, \beta_i]\}, for\ i = 1, 2, ..., N$. The choice of DOE is up to the user's preference, and will most likely problem dependent. This initial DOE is used as the training data for the Random Forest model, and thus all design points in the training DOE must be classified as either a feasible design (1) or infeasible design (-1). This classification incorporates any of the constraints discussed in Section 2.1, and at this point the classification value represents the global classification of each design point (i.e. any design classified as feasible meets ALL of the constraints that were applied to the system. If ANY constraint is violated, then the design point is to be classified as infeasible). It is possible to increase the complexity of the SeBBAS algorithm by considering each constraint separately, but this capability must be implemented in future versions of the code.

Once classified, a RF model is fit to the training data, and the iterative process of learning the constraint boundaries and refining the RF model begins. This process is carried out in the following steps:

1) Fit a Random Forest model to the classified training data set
2) Validate the RF model against a validation data set
3) Based on the RF model and validation data set, suggest new refinement data points by selecting randomly sampled points that have the highest probability of being incorrectly classified (these are the blue points in Figure 2-2)
4) Classify the refinement data points, and add them to the training data set
5) Repeat steps 1-4 until max number of iterations reached, error tolerance has been met, or all resources have been used

Once the RF model has been fit and refined, it serves two purposes. First, it can be used to suggest new design points to sample that fall within the feasible region of the design space. This is beneficial when a large portion of the design space is infeasible due to constraints, as it will allow you to densely sample the feasible design region of the design space to create a DOE for fitting surrogate models. Once the surrogate models have been created, the RF model can be used to check that any design point run through the surrogate models are actually feasible designs. As discussed in Section 2.2, this is a necessary feature as it prevents the surrogate models from extrapolating when a design point is actually infeasible.
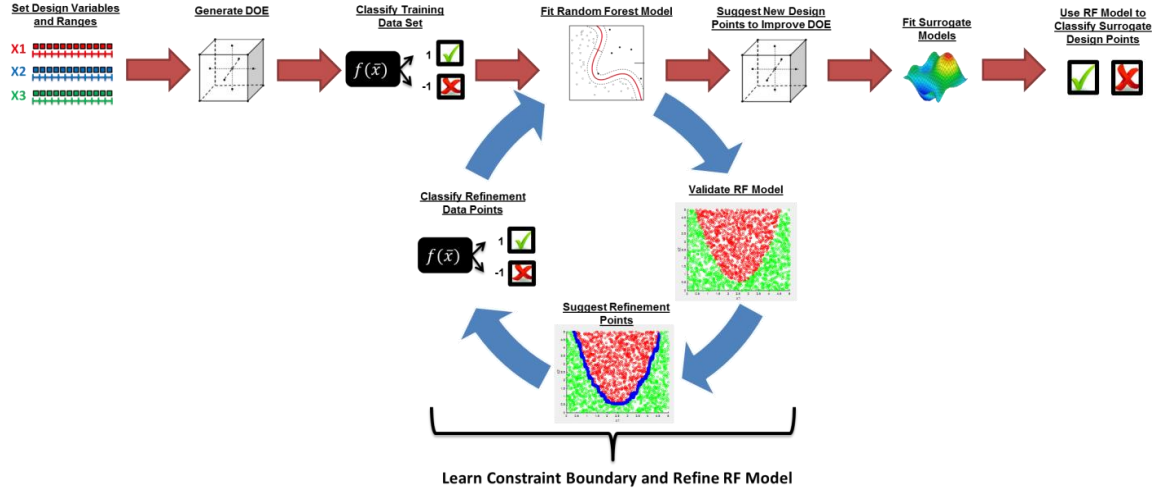
Figure 2-2: SeBBAS approach used to generate a Random Forest model capable of classifying design points as feasible or infeasible based solely on the design variable values

2.4   SeBBAS Implementation

Currently, the SeBBAS algorithm is implemented using a combination of MATLAB and the R statistical programming language. The user interacts only with MATLAB, which requires a single script file to set up the proper inputs and call the SeBBAS algorithm. The remainder of the algorithm is entirely automated, although the interfacing between MATLAB and R does require that a number of folders and files are located in the proper directory on the local computer. All actions required by the user to properly set up and run the SeBBAS algorithm with an arbitrary objective function are discussed in the remainder of this section.

2.4.1   SeBBAS Files & Folder Structure

There are two main files that are required in order to run SeBBAS: a MATLAB function called *SeBBAS.m*, and a R script called *SeBBAS.R*. These two files must be located in the same directory location on the local computer. In addition, a folder called *R Files* must be located in the same directory location as the two SeBBAS scripts. This folder contains the files required for MATLAB and R to interact with one another. An example of this file structure is shown below in Figure 2-3. In Figure 2-3, the additional MATLAB files *run_Test_SeBBAS.m* and *SeBBAS_Tests.m* are simply additional MATLAB scripts and functions used to test the SeBBAS algorithm. For convenience, they are placed in the same directory location as the SeBBAS algorithm.

The *R Files* folder contains a series of additional folders that are used to organize the various input and output files required in the SeBBAS algorithm with its current setup. These folders are shown in Figure 2-4, and the important folders are described in the proceeding subsections. In addition to these folders, the *SeBBAS_Input_Files.csv* file must be located in the *R Files* folder. This .csv file contains the inputs required

19

to properly run the *SeBBAS.R* script, and is automatically written by the MATLAB *SeBBAS.m* function. It should be noted that the spelling of these folders MUST be matched exactly, as both the *SeBBAS.R* and *SeBBAS.m* functions look for files in these specific folders, which is what allows the process to be automated.

Figure 2-3: SeBBAS file locations relative to one another on local computer

Figure 2-4: Folder structures that contain inputs, outputs, and results from SeBBAS algorithm

SeBBAS Input File

The SeBBAS_Input_File.csv is read directly by the SeBBAS.R script. Currently, there are 10 inputs that are required in the input file, which are described in Table 2-1. The input file should contain 10 columns, one for each of the ten input variables, and two rows. The first row contains the variable name, which must be spelled exactly as shown in Table 2-1, and the second contains the value assigned to each variable. Again this process is completely automated in MATLAB, and is just listed here for reference.

Table 2-1: Required inputs of the SeBBAS_Input_File.csv file

Input Parameter Name	Data Type	Description
num_DV	Integer	The number of design variables used in the current DOE
AS_budget	Integer	The adaptive sampling budget (i.e. the number of new design points that can be run through the objective function) available for the current run.
AS_DS_factor	Integer	The adaptive sampling – dense sampling factor. This determines how many new design points will be randomly sampled and classified using the RF model.

AS_option	String	• Refine: the SeBBAS.R algorithm will refine the RF models fit of the constraint boundary by suggesting new points to sample along the boundary. • Sample: the SeBBAS.R algorithm will suggest new points to include in a DOE that will have a high probability of being feasible designs
Classified_DOE_Filename	String	The .csv filename for the classified DOE, located in the *Classified DOEs* folder.
RF_Fit_SaveAs	String	The name that the current RF model is saved as. The variable should have a .rda file extension, and is saved in the *RF Fits* folder
DV_Range_Filename	String	The name of the .csv file that contains the name of each design variable and corresponding min and max allowable value. Located in the *Design Variable Range* folder.
Validation_DOE_Filename	String	The name of the .csv file that contains the validation classified DOE. This file is also located in the *Classified DOEs* folder.
Output_DOE_Filename	String	Contains the file name used to output either a DOE of design points used to refine the RF model, or a DOE to suggest feasible design points to sample. If "AS_option" is set to "refine", then the file will be saved in the *Refinement DOEs* folder. If it is set to "sample", it will be saved in the *Suggested Sample DOEs* folder
Variable_Importance_SaveAs	String	Filename used to save the information on the importance of each variable in the RF model. This .csv file is saved in the *Variable Importance* folder.

Classified DOE's

The term "Classified DOE" will appear repeatedly in the remainder of this report. A classified DOE simply refers to a .csv file that consist of a set of design points that have been run through the objective function and classified as either feasible (value of 1), or infeasible (value of -1) based on the constraints provided within the objective function. The Classified DOE.csv file will have N+1 rows and M+1 columns, where N is the number of design points or cases, and M is the number of design variables. The extra row and column account for the following:

- Column 1: The classification of each design point as either feasible or infeasible
- Columns 2 – (M+1): The values of the design variables for the current design point
- Row 1: The header row that contains "Classification" (or some other term that you wish) in column 1, and the names of the design variables in the remaining columns
- Rows 2 – (N+1): The classification and design variable values for each design point

To make things clearer, an example is provided in Figure 2-5 for a system with two design variables. In row one, the column headers are provided, where "Global Feasible" is used as the header to the classification column, and the two design variables are labeled "x1" and "x2". Starting in row 2, the classification of each design point is listed, as well as the corresponding values for each design variable. This is repeated for each design point in the DOE. This example can easily be extrapolated for systems with any number of design variables by just adding the classification value in column one, followed by the values of each design variable in the proceeding columns.

	A	B	C
1	Global Feasible	x1	x2
2	-1	4.142878	4.039256
3	-1	0.042256	1.245576
4	-1	4.754379	4.791714
5	-1	1.639149	0.966297
6	-1	1.396814	4.558148
7	-1	3.318521	4.86809
8	1	2.683393	0.878561

Figure 2-5: Example of a "Classified DOE" on a system with two design variables

The Design Variable Range folder contains .csv files with information on each design variables name, along with their min and max allowable values. An example of this is shown below in Figure 2-6. This example has four different design variables, which are labeled in row 1. The minimum and maximum value that each design variable can have are listed in rows 2 and 3, respectively.

	A	B	C	D
1	x1	x2	x3	x4
2	0	-5	1	-10
3	5	5	6	0

Figure 2-6: Example .csv file for setting design variable ranges

These two folders contain a DOE or rather a list of new design points that should be sampled and classified using the objective function. The *SeBBAS.R* script writes .csv files to these folders to pass information back to MATLAB. The format of the DOE files are almost identical to the classified DOE shown in Figure 4-3, with the exception that the classification column is no longer present. An example of this is shown in Figure 4-5, where the design variable names are listed in row 1, with the new design points in the remaining rows.

	A	B
1	x1	x2
2	4.378866	0.053681
3	3.804912	-1.74795
4	4.430623	-0.97884
5	1.625477	-4.47053

Figure 2-7: Example of the DOE format required to suggest new design points to sample

2.4.2   Setup of R Programming Language for SeBBAS

Software Installations Required

Two separate software packages associated with R must be installed to run SeBBAS. First, R must be installed, which contains the base binary files required for R along with the randomForest Software package. This program can be downloaded from https://www.r-project.org/. Secondly, RStudio software must be downloaded through ANACONDA, which is an IDE for Python v3.6 and R. ANACONDA v4.3.1 or later must be downloaded from https://www.continuum.io/downloads. Once ANACONDA is installed, open the ANACONDA navigator and install RStudio through ANACONDA (this may require admin access). The ANACONDA RStudio package contains the Rscript executable file which allows R to be called through the command prompt, which is key to automating the SeBBAS process. Once both R and RStudio are installed there are two remaining steps before R has been setup properly to run for SeBBAS.

Setting Windows Path Environment Variable for Rscript

MATLAB calls the *SeBBAS.R* script by using the command *system('Rscript SeBBAS.R')*. This command requires that a windows PATH environment variable exists for Rscript. Generally this is done at the installation of ANACONDA. However, if the error message shown in Figure 2-8 occurs in MATLAB when SeBBAS is run, then the PATH environment variable was not correctly set and must be done manually.
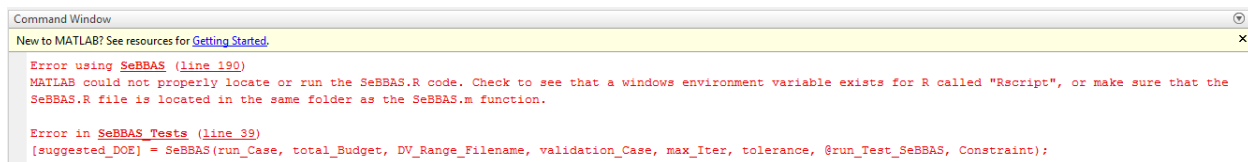
Figure 2-8: Error exception thrown in MATLAB if Rscript windows environment variable has not been created for R

To set the windows PATH environment variable (obviously this only works for Windows machines), first locate the *Rscript.exe* file by searching for it on your computer through the start menu. It should be installed as a script file of ANACONDA, and thus located in a subfolder of the ANACONDA program. An example of how this might look is shown in Figure 2-9.
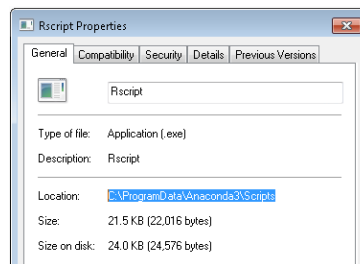
Figure 2-9: One example for the likely location of Rscript.exe file on local computer

Once the location of the *Rscript.exe* file has been located, navigate to "Control Panel -> System -> Advanced System Settings" on the local computer (or type "System Environment Variables" into the start menu). This will bring up the window shown in Figure 2-10. If not already selected, navigate to the "Advanced" tab, and select the Environment Variables…" button in the bottom right. In the *Environment Variables* window, go to the *System Variables* scroll box, select the *Path* (or *PATH*) system variable and click "Edit…". In this new window (shown in Figure 2-11) append the location of the Rscipt.exe file from Figure 2-10 to the variable value and hit enter. Not only the folder location is required (i.e. end at "Scripts" as shown in Figure 2-11), as windows will automatically search for any .exe file located in this path to find the *Rscript.exe* file. This will successfully add the *Rscript.exe* file to the windows Path environment variable.
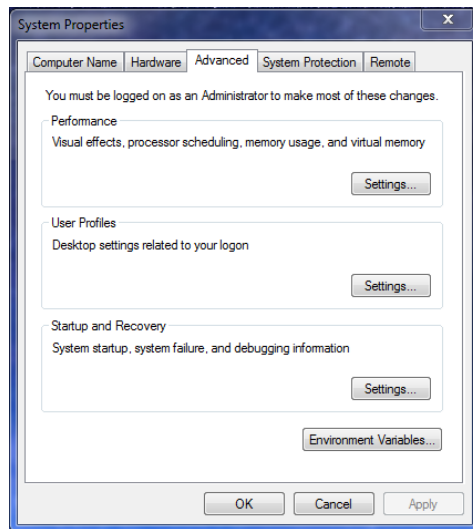
Figure 2-10: System Properties window used to edit environment variables
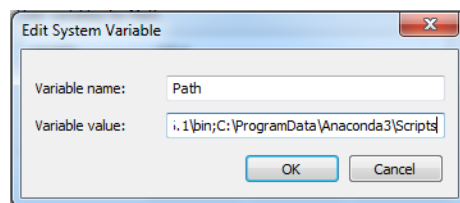
Figure 2-11: Adding Rscript file location as a windows environment variable

Installing randomForest package in R

An additional requirement of SeBBAS is that the randomForest package be installed in R. In Windows, this can be done by opening the R program and using the toolbar to navigate to "Packages -> Install Package(s)…". When the "HTHTPS CRAN mirror" window pops up, select "0-Cloud [https]" and click OK. In the Packages window, select randomForest and click okay again to load it. It should be noted that if R was installed as an administrator, then R must be launched as an administrator before attempting to

load the randomForest package. If it has not been installed, then the error message shown in Figure 2-12 will be shown in MATLAB when SeBBAS is run. If these instructions do not work, then please consult the internet for help.
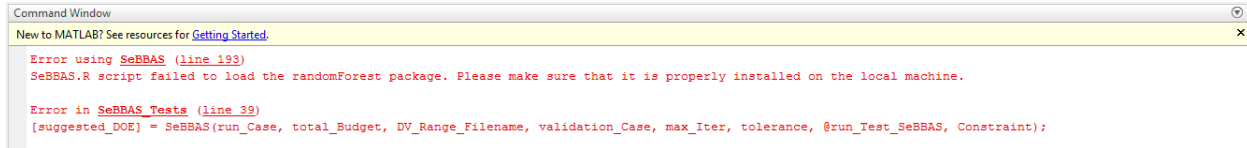
Figure 2-12: Error exception thrown in MATLAB if randomForest package is not properly installed in R

Modification Required to SeBBAS.R Script

One modification must be made by the user to the *SeBBAS.R* script before the process can be automated. Because the R programming language does not have the ability to determine the local directory or path of the script itself, the user must manually set this. To do this, open the *SeBBAS.R* script, and on (or around) line 111 of the code, the "parentDirectory" variable must be set to the path of the *SeBBAS.R* script. For example, if the *SeBBAS.R* script is located in the folder "SeBBAS Code", which is located on the Desktop of the local computer, then the parentDirectory variable must be set as shown in Figure 2-13. If the user fails to set the parentDirectory variable correctly, then MATLAB will throw an exception shown in Figure 2-14 when run.

Figure 2-13: Example of how to properly set the parentDirectory variable in the SeBBAS.R script
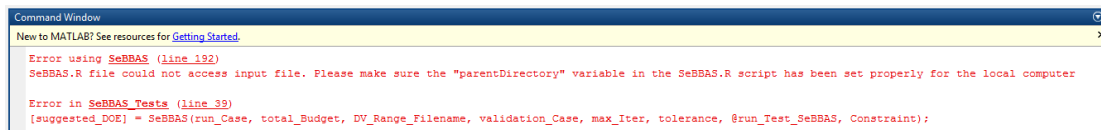
Figure 2-14: Error thrown in MATLAB when the parentDirectory variable in the SeBBAS.R file is not set properly

2.4.3   Running SeBBAS Algorithm in MATLAB

The user runs the SeBBAS algorithm in MATLAB by creating two additional files. First, a script file must be created to set the necessary input variables, and call the *SeBBAS.m* function. The second file is a function used to evaluate the objective function of the current study. The format of both of these files are discussed in the following subsections.

25

The SeBBAS.m function in MATLAB requires eight inputs to run properly, all of which are described in Table 2-2. All of these input parameters are straight forward with the exception of the Obj_function variable. In MATLAB, an objective function must be created that takes the design variable as inputs, and returns a classification of either feasible or infeasible (this will be discussed in more detail in Section 2593088.0.-946779886), which is simply a function in MATLAB. This function can be passed to the SeBBAS algorithm as a function handle by placing the @ symbol in front of the Obj_function name. For example, if the objective function created in MATLAB is called "Constraint_Test1", then this would be passed to the SeBBAS function as a function handle by entering *SeBBAS(@Constraint_Test1)* when calling the SeBBAS.m function. 4

Table 2-2: Input parameters required for the SeBBAS algorithm

Input Parameter Name	Data Type	Description
run_Case	String	String used to identify the current run in SeBBAS. All .csv files will be saved as an extension on this name. For example, the .csv file for refinement will be saved as "run_Case_Refinement.csv".

total_Budget	3x1 int Cell Array Or 3x1 {String, int, int} Cell Array	Three values are expected in this input:
		[1] Initial Sample Budget:
		  a. Int: the number of design points that will be used to create the initial DOE to sample the design space
		  b. String: the name of the .csv file that contains a pre-classified initial sample DOE
		[2] Refinement Budget: the max number of function calls that can be made to the objective function when refining the RF model to identify the constraint boundary
		[3] Suggested Sample Budget: the minimum number of design points that will be sampled to suggest feasible design points to fit a surrogate model to

DV_Range_Filename	String	The name of the .csv file that contains the variable names and the min and max value of each variable
validation_Case	String or Integer	• **String**: If a string data type is provided, it must contain the name of the .csv file that contains the classified validation DOE. This option should be used if the objective function is expensive to evaluate, enabling the validation DOE to be pre-calculated to reduce run time. • **Integer**: If int data type, the SeBBAS algorithm will classify an additional DOE with the number of design points equal to the value provided by this variable. This option should be used if the objective function is

26

		not expensive to evaluate, as it reduces the work that the user must do.
max_refinement_Iter	Integer	The max number of iterations that the SeBBAS algorithm will perform to refine the RF model.
Tolerance	Double	Stopping criteria that will stop the refinement of the RF model if the percentage of incorrectly classified validation data points falls below this tolerance.
Obj_function	Function Handle	A function handle that is passed to the SeBBAS algorithm that has the structure as described in Section 2593088.0.-946779886.
Add. Inputs	Structure	If the objective function requires any other inputs to run properly, then all of those additional inputs must be packaged in this single parameter. If multiple additional input parameters are required, they can be packaged into a single structure variable in MATLAB, then unpackaged in the Objective Function code for use. If no additional inputs are required, then omit this parameter from the function declaration.

Objective Function Structure

The objective function of the design problem must be written as a MATLAB function, with the inputs and outputs as shown in Figure 2-15. It is passed to the SeBBAS algorithm as a function handle, as described in Section 2593088.0.-946779886. The objective function takes a set of design points (called a DOE in this case), and classifies each design point as either feasible or infeasible. If the objective function created in MATLAB requires additional inputs beyond the design variable values, then they must all be passed to the objective function in a single variable. Each of the input and output parameters of the objective function are described in detail in Table 4-3 and Table 4-4, respectively.
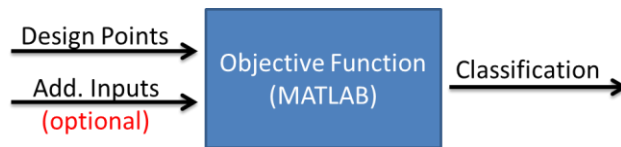
Figure 2-15: Structure of objective function in MATLAB

Table 2-3: Description of all input parameters required for the Objective Function

Input Parameter Name	Data Type	Description
Design Points	NxM Double Array	A DOE or set of design points to run through the objective function. The columns of the array correspond to the M design variables,

		while the rows of the array correspond to different design points being evaluated.
Add. Inputs	---	If the objective function requires any other inputs to run properly, then all of those additional inputs must be packaged in this single parameter. If multiple additional input parameters are required, they can be packaged into a single structure variable in MATLAB, then unpackaged in the Objective Function code for use. If no additional inputs are required, then omit this parameter from the function declaration.

Table 2-4: Description of all output parameters required for the Objective Function

Output Parameter Name	Data Type	Description
Classification	Nx1 Integer Array	Returns an array of 1's and -1's, where values of 1 correspond to a feasible design, and values of -1 correspond to an infeasible design.

If an objective function only requires the values of the design variables as inputs, then the "Add. Inputs" variable should be omitted from the function itself. If additional inputs are required (i.e. the constraints are inputs to the function) then include this variable. If multiple additional inputs are required, these can be packaged as a MATLAB structure or array, and unpackaged in the objective function. An example of this is shown in Figure 2-16, where constraints on the values of x1 and x2 are packaged into a single structure variable in MATLAB, and then unpackaged inside the objective function. Regardless of what approach is taken, the user must write the objective function, and will thus know how to package and unpack these additional inputs, which will likely vary on a case by case basis.

```
x1_upper_limit = 5;
x2_lower_limit = 2;

% Packaging the two constraints into a single structure variable
add_input.x1_upper_limit = x1_upper_limit;
add_input.x2_lower_limit = x2_lower_limit;

[Classification] = function objective_function(DOE, add_input)

% x1 variables are column 1, x2 variables are column 2
x1 = DOE(:,1);
x2 = DOE(:,2);

% Unpackaging the add_input to pull out the two constraints
x1_upper_limit = add_input.x1_upper_limit;
x2_lower_limit = add_input.x2_lower_limit;

% Applying the constraints to x1 and x2
if x1 > x1_upper_limit || x2 < x2_lower_limit
    Classification = -1;
else
    Classification = 1;
end
```

Figure 2-16: Example showing one possible way to package additional inputs into a single variable

2.5   Test Cases for SeBBAS

A series of 2D and 3D test functions were created in MATLAB to test the SeBBAS algorithm. The test objective function and the script set up to run SeBBAS with the objective function are listed in Appendix A. Ten different constraints are available in this test objective function, listed below. A few of them are tested and discussed here. For the purposes of this report, the SeBBAS algorithm was slightly modified so that the points suggested to refine the RF model will appear as blue squares at each iteration.

1) Infeasible cross in middle, creating 4 squares in each corner
2) Upper triangle of hypercube is infeasible
3) Parabolic constraint
4) Points that fall within circle centered at (2.5, 2.5) are infeasible
5) Hyperbolic constraint
6) Triangular constraint in the middle of the hypercubic design space
7) Linear band with slope 1 across hypercubic design space is infeasible
8) Cubic constraint on x1 variable
9) Sharp corner constraint boundary
10) 3D sphere of feasible design points

2.5.1   Parabolic Constraint Test

The parabolic constraint (constraint 3) was tested on the domain $D = \{x_1, x_2 \in [0,5]\}$, using the following sampling budget:

- Initial DOE Sample: 50 design points
- Refinement Budget: 2500 design points

The progression of the refinement points being sampled to learn the constraint boundary is shown in Figure 2-17. The RF model is initially fit to the Initial DOE Classification data (top left figure), which is a fairly sparse sample of the design space. The RF model then starts to sample design points that it believes to lie near the boundary. At iteration 1, the suggested refinement points resemble a box. SeBBAS then classifies these refinement points as either feasible or infeasible designs, which can easily be seen if Iteration 2 is observed (points that were blue in Iteration 1 have now been classified as feasible or infeasible and plotted again). As the RF model learns more about the design space, the suggested refinement points begin to closely resemble the parabolic constraint put into place. As expected, the training data has a dense sample of points near the constraint boundary, which helps the RF model refine this boundary as accurately as possible.
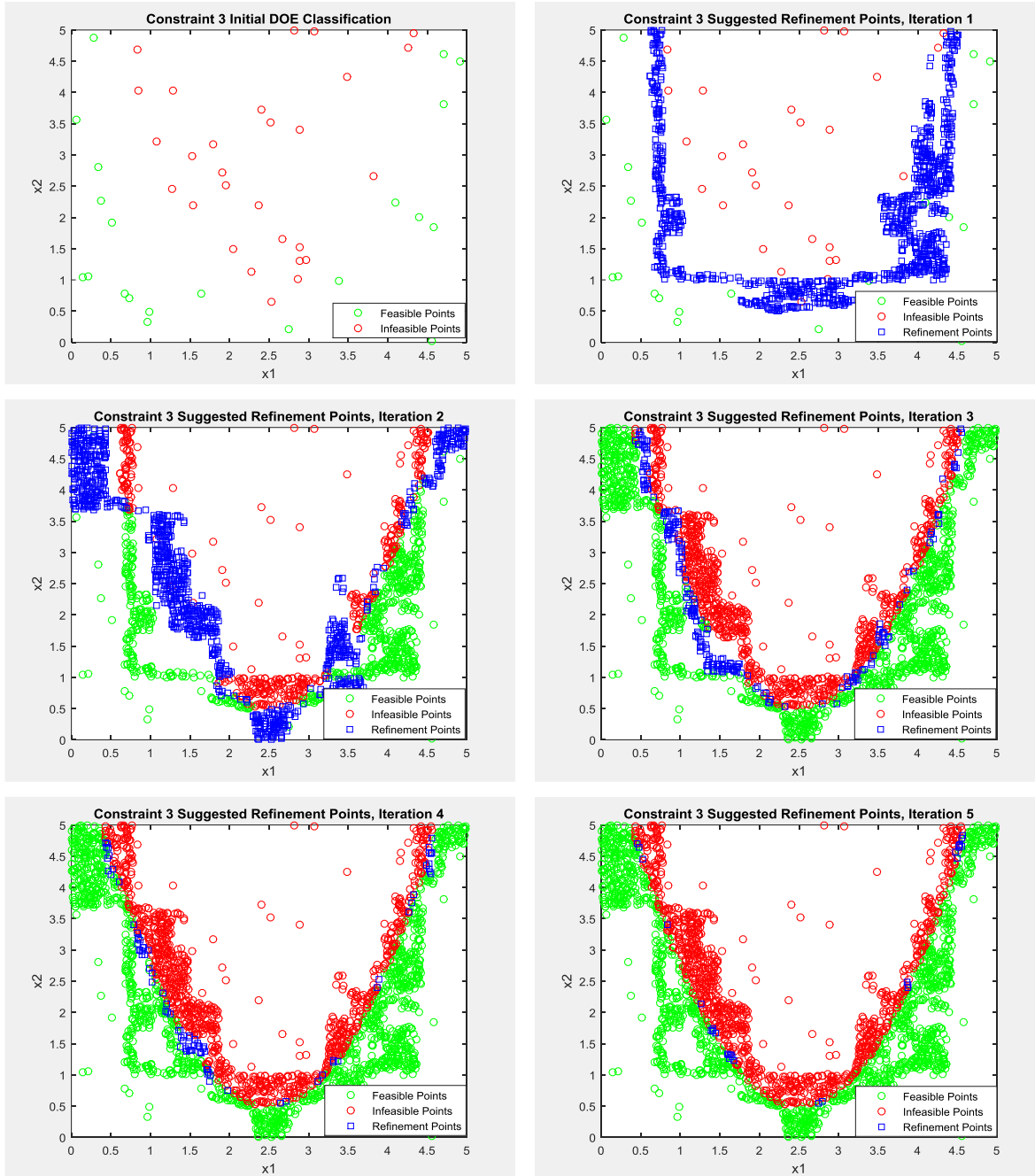
Figure 2-17: Progression of RF algorithm sampling points to learn the location of a parabolic constraint boundary

After Iteration 5, the RF model was used to suggest approximately 12,500 design points that it classified as feasible. Of these 12,500 design points, only 0.72% of them were classified as false positives (a false positive is a design point that the RF model classified as feasible, when in reality it was infeasible). These results, shown in Figure 2-18, indicate that the RF model is now an extremely accurate classification tool.
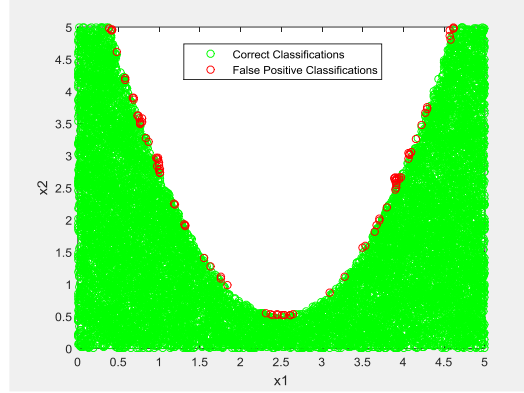
Figure 2-18: Results of RF classification model for parabolic constraint

2.5.2 Sharp Corner Constraint Test

A second example was performed using the sharp corner constraint (Constraint 9). The initial DOE and refinement sample budgets remained the same, but this time a domain of $D = \{x_1 \in [0,5], x_2 \in [-5,5]\}$. This test was used to determine if SeBBAS and the RF model could resolve a boundary with a sharp corner. The results, shown in Figure 2-19 and Figure 2-20, indicate that the test was successful.
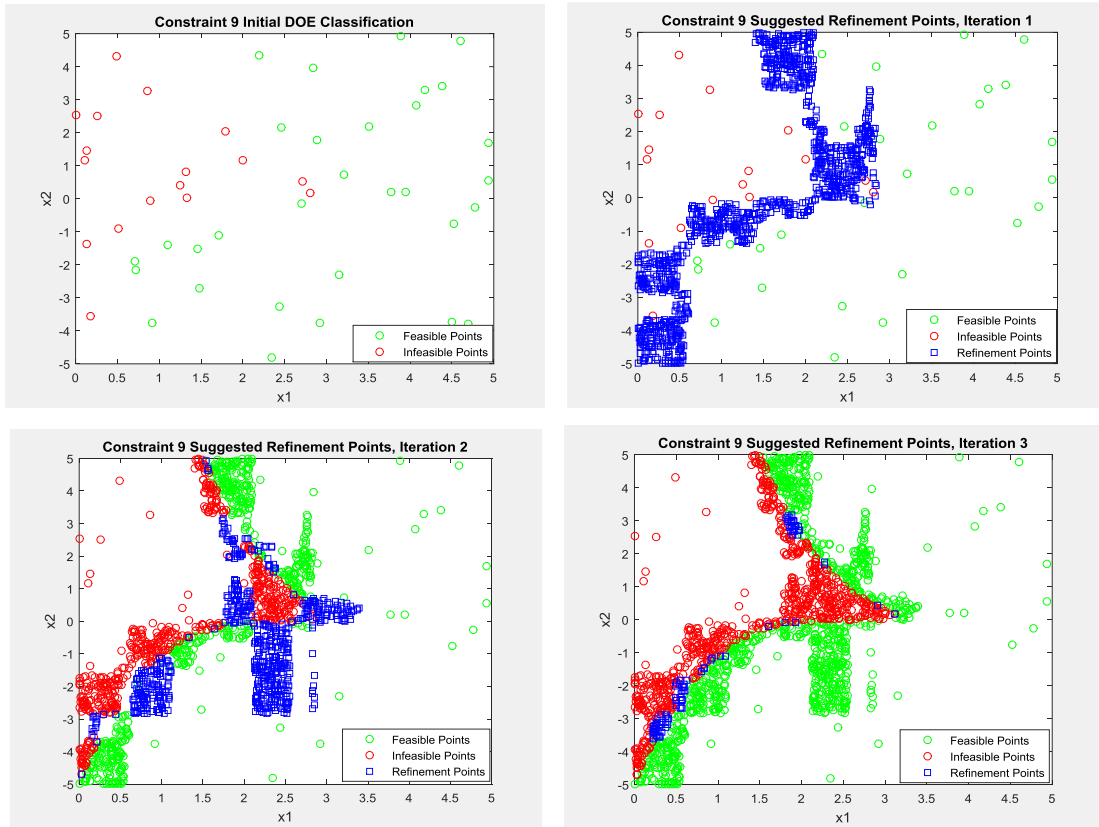
Figure 2-19: Progression of RF algorithm sampling points to learn the location of a sharp corner constraint boundary
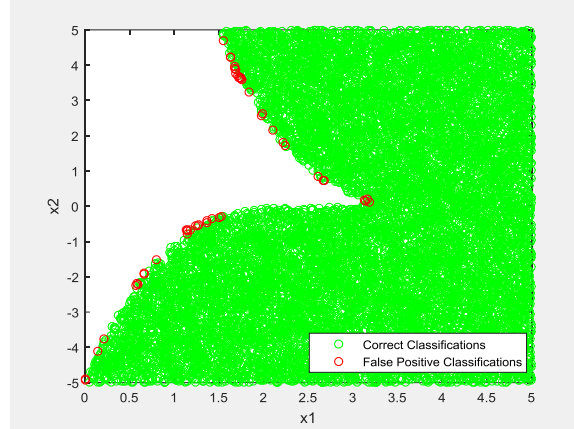
Figure 2-20: Resolution of RF algorithm on corner constraint boundary

2.5.3   Significance of Test Results

The process of training a Random Forest model to learn an arbitrary constraint boundary can be a rather expensive process. In addition to running the training data set through the objective function, the SeBBAS approach also requires that a validation data set be classified so that the accuracy of the RF model can be tested. Depending on the number of dimensions and complexity of the design space, it could potentially take thousands of refinement data points to refine the RF model to a sufficient accuracy. Almost certainly this process will be far more computationally expensive than simply running a structured, hypercubic based DOE and fitting a surrogate model to the feasible design points.

However, as discussed earlier a huge disadvantage of this traditional surrogate model approach is that it has no way of identifying infeasible versus feasible design points. If a point was sampled in the infeasible region of the design spaces from Figure 2-18 or Figure 2-20, the surrogate model will extrapolate (this is an extrapolation because no information from these portions of the design space were used to construct the surrogate model). In this extrapolation zone, the results of the surrogate model become completely unreliable. In the academic examples presented in this section, it would be easy for the user to anticipate these infeasible zones in the design space, but in more complex problems this is not a viable option. In addition, in more complicated functions the infeasible zones will likely be due to a code failing or not converging due to some underlying physical constraint, rather than artificial constraints applied after the fact. Because of this, the ability to create a classification model of the design space becomes a huge advantage as the complexity of the problem increases.

2.6   NDARC Test with SeBBAS

The objective of this study was to implement the SeBBAS algorithm using NDARC as the objective function. To accomplish this, a MATLAB function was created following the requirements outlined in

32

Section 2593088.0.-946779886. This function fully automates the process of writing the input files, running NDARC and parsing the NDARC output to extract the desired performance metrics. A brief overview of how this was accomplished is presented in Section 2.6.1. The results of the NDARC study with SeBBAS are presented in Section 2.6.2.

2.6.1 MATLAB Function for NDARC

A MATLAB function called *NDARC.m* was created to allow SeBBAS to be used with NDARC. In addition to the design variable values from the DOE, this function also requires the NDARC variable names corresponding to each design variable. This is because the NDARC variable names are used to write the design variable values to the correct location in the NDARC input files, as will be discussed in Section 2593088.0.-946779886. Thus, as outlined in Table 2-2, the *Add. Inputs* input variable to the SeBBAS algorithm should include the NDARC variable names corresponding to each of the design variables.

Figure 2-21: Inputs required to run the NDARC.m MATLAB function

There are three main steps required to run NDARC from MATLAB. First, the NDARC input files must be written, while taking into account changes to any design variables from the current DOE being run. To accomplish this, template NDARC input files are stored as MATLAB cell arrays that can be modified before rewriting the NDARC input files in the proper format for NDARC. This process is discussed in Section 2593088.0.-946779886. Next, the NDARC itself must be run, which is outlined in Section 2593088.0.-946779886. Finally, performance metrics must be extracted from the NDARC output file, which is discussed in Section 2593088.0.-946779886.

NDARC File Structure

The *NDARC.m* function must be located in the same folder as the *SeBBAS.m* function to run properly. In addition, a folder was created to store all NDARC related files, called *NDARC Files*. This folder must be located in the same folder as the SeBBAS code, and the MATLAB function used to call NDARC. The locations of the *NDARC.m* MATLAB function and the *NDARC Files* folder relative to the *SeBBAS.m* function are shown in Figure 2-22.

Figure 2-22: File and folder locations required to properly run NDARC

The required contents of the *NDARC Files* folder are shown in Figure 2-23. First, the NDARC executable must be located in this folder. The current *NDARC.m* function is hard coded to call the *ndarc_1_9_x64.exe* version of the code, so if a different version of NDARC is used then the MATLAB function must be changed, as outlined in Section 2593088.0.-946779886. The *Size.njob* file is the main input file for NDARC, which tells NDARC where all of the other input files are located on the local machine. These "other" input files (which will be discussed in Section 2593088.0.-946779886) are all located in the *NDARC Inputs Files* folder. The *SizeOutput* folder contains all of the output files written by NDARC. Finally, the NDARC Template Files folder contains the MATLAB versions of the NDARC input files, which allow the design variables to be changed before running NDARC.

Figure 2-23: Required contents of NDARC Files folder

Writing NDARC Input Files

In general NDARC requires six input files to run, which are described in Table 2-5 below. The *Size.njob* file tells NDARC the location of these six input files, which for SeBBAS is the *NDARC Input Files* folder as shown in Figure 2-23. For this study, it was assumed that the sizing and solution settings would remained fixed for each design point. The other four input files all contain NDARC parameters that could potentially be treated as design variables. Thus, an approach was developed that would allow each of these inputs to be varied to account for changes in the desired design variables from some baseline helicopter model.

34

Table 2-5: Input files required to run NDARC

Input File	File Extension	Description
Engine Model	.list	Contains inputs for the performance and specifications of the engine.
Flight Conditions	.cond	Inputs for the flight conditions (i.e. altitude, velocity, etc.)
Mission Statement	.miss	Inputs that specify the mission(s) that the helicopter will be sized to.
Aircraft Configuration	.airc	Inputs that define the configuration of the helicopter.
Sizing Settings	.size	Specifies the reference points that will be used to size the helicopter
Solution Settings	.sol	Sets the parameters used for the internal solvers in NDARC, such as max number of iterations and relaxation parameters.

Constructing Template Input Files

The approach taken was to save a baseline input file for the engine model, flight conditions, mission, and aircraft configuration as a template file. To do this, the baseline input files were imported to MATLAB and saved as cell arrays. The cell arrays re saved as .mat files in the *NDARC Template Files* folder shown in Figure 2-23. Within each cell array, the location of the desired NDARC variables were identified (the location that they had within the cell array is given as some combination of a row and column number), and hardcoded into the *writeNDARC_InputFiles* subfunction of the *NDARC.m* function. This subfunction in turn matches the provided NDARC variable name to the correct template file, writing each design variable to the correct location. This is why the design variable names are required ad an additional input to the *NDARC.m* function. An example of how this works is shown in Figure 2-24, where the list of variable names is compared to an available list of NDARC variables. If a match is found, the corresponding design variable will be written to the correct location in the correct template file. If no match is found, then an error will be thrown, halting the execution of SeBBAS and indicating to the user that this variable must be added.

```
switch VariableNames{k}
    % Writing to the aircraft configuration template file
    case 'diskload'
        diskload = DesignVariables(k);
        UH60_airc_InputTemplate{243,8} = strcat(num2str(diskload), ',');
    case 'CWs'
        CWs = DesignVariables(k);
        UH60_airc_InputTemplate{244,8} = strcat(num2str(CWs), ',');
```

Figure 2-24: Example of how NDARC function uses variable names to write design variable values to correct locations

To maintain consistency in variable names, the list of NDARC variable available in the *NDARC.m* MATLAB function are identical to the names that are provided in NDARC's Input Manual, with one

exception. In some instances (mainly with the drag coefficient parameters), a NDARC variable has the same name across different structures. For example, the variable name for zero lift drag coefficient, CD, is the same for the fuselage as it is for the wing or tail. In this case, the variable is given an extension to specify where to write it in the input template files. For the fuselage, CD would be given the extension "_fus", making it CD_fus. The NDARC variable that are currently available in the *NDARC.m* for function are listed in Appendix B, with descriptions provided to increase their clarity.

It should be noted that the input template files must be altered every time a new study is performed using a different baseline helicopter as the starting point. It is possible to directly edit the .mat template files to enter the new values for the baseline helicopter, in which case the NDARC variable locations will remain the same. The other option is to import the baseline helicopters input files and create new templates directly from them. In this case, the user would have to rewrite the *writeNDARC_InputFiles* in the *NDARC.m* MATLAB function to specify the new location of each NDARC variable. Unfortunately, to the best of this author's knowledge there is no easy way to get around this. However, once the template files have been created and the *writeNDARC_InputFiles* files adjusted, they will be ready for use with SeBBAS.

Running NDARC

Because the NDARC executable is run through the command prompt, it must be called using the system command in MATLAB. This is performed in the *runNDARC* function embedded within the *NDARC.m* function. The NDARC executable requires two inputs at the command line, the filename that contains information on where all other NDARC input files are located (this is the *Size.njob* file), and the filename to write the general results to (called *UH60.out* for this study). The command to call NDARC itself relies on knowing the exact name of the NDARC executable itself. For this study, the NDARC executable used is called *ndarc_1_9_x64.exe*. The command to call NDARC is:

> 'ndarc_1_9_x64.exe <Size.njob> .\SizeOutput\UH60.out'

The first entry after the NDARC executable is the input file name located within <> symbols. The next entry is the desired path location and filename of the output file. For the example shown above, NDARC will read the *Size.njob* file (which must be located in the same folder as the NDARC executable) to determine the location of all of the other input files, and it will write the *UH60.out* output file to the *SizeOutput* folder.

Extracting NDARC Performance Metrics

Currently, the NDARC output file is read using MATLABs textscan function, which reads the output file information into a cell array. The values of the desired performance metrics are then extracted from this

cell array from their known locations within the cell array. To determine the locations of a given performance metric, the MATLAB cell array was searched once to find the location of each performance metric, and their locations were hard coded into the MATLAB function. Several tests were run, and it was found that the location of each performance metric remained the same as the design variables were changed. Admittedly, this is not a very robust approach, as it is not known if the location of the performance metrics within the cell array will vary on rare (or possibly even more common) occasions that were not tested for. However, the current study did not rely on extracting the correct values for certain performance metrics, but rather if NDARC converged or not. If NDARC failed to converge, then NAN would be returned in place of the first performance metric value.

In future studies, the performance metrics from NDARC will likely be used as constraints. Because of this, a more robust approach to parsing the NDARC output file should be implemented in place of the current *readNDARC_Output* function. The constraints must then be written into the *NDARC.m* function that will classify design points as either feasible or infeasible based on the current designs performance. Current classification is based solely on whether NDARC was able to converge to a solution for a given design point, taking no other constraints into consideration.

Summary of Running NDARC

The general process discussed above is outlined in Figure 2-25. Any time a new design point needs to be tested, the NDARC design variables will be written to the correct location within the input files. NDARC will then be run using the MATLAB system command. If NDARC is successfully executed, then the current design point will be classified as either feasible or infeasible.
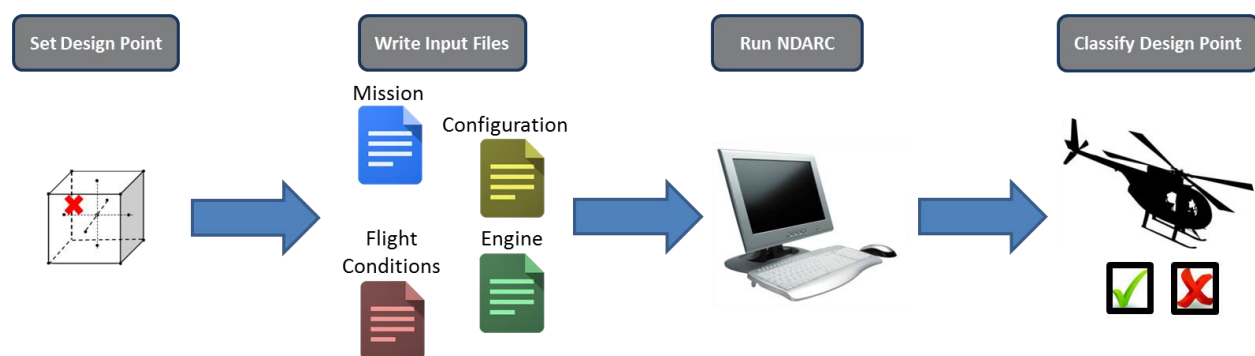
Figure 2-25: Summary of process used to classify a NDARC design point

2.6.2　NDARC Case Study

In the test cases studied in Section 2.4 the SeBBAS approach provided incredibly accurate classification models, with results showing an accuracy rating of over 99% for the 2D test cases. Based on these initial results, the following hypothesis was proposed:

If the RF classification model offers a significant improvement over the current surrogate model approach, a significantly higher percentage of feasible points would be sampled through the use of this RF classification model than would be sampled using randomly design points across the entire design space.

To test this hypothesis on an engineering design problem, a case study was carried out using a UH-60 helicopter as the baseline model. First, a RF model was fit to NDARC using the process outlined in Figure 2-2. Two separate tests were then carried out, as described by Figure 6-6. First, a set of 1000 design points were randomly selected and classified across the entire hypercubic design space. This serves as a baseline value to approximate the percentage of the design space that is infeasible. Next, the RF classification model was use to suggest 1000 design points that it had already classified as feasible. The percentage of infeasible design points sampled between these two tests are then compared. If the hypothesis is correct, then the RF classification model offers a significant improvement to the current method, as it would be capable of accurately classifying design points that are tested through surrogate models without requiring the user to run each design point through an expensive engineering model.
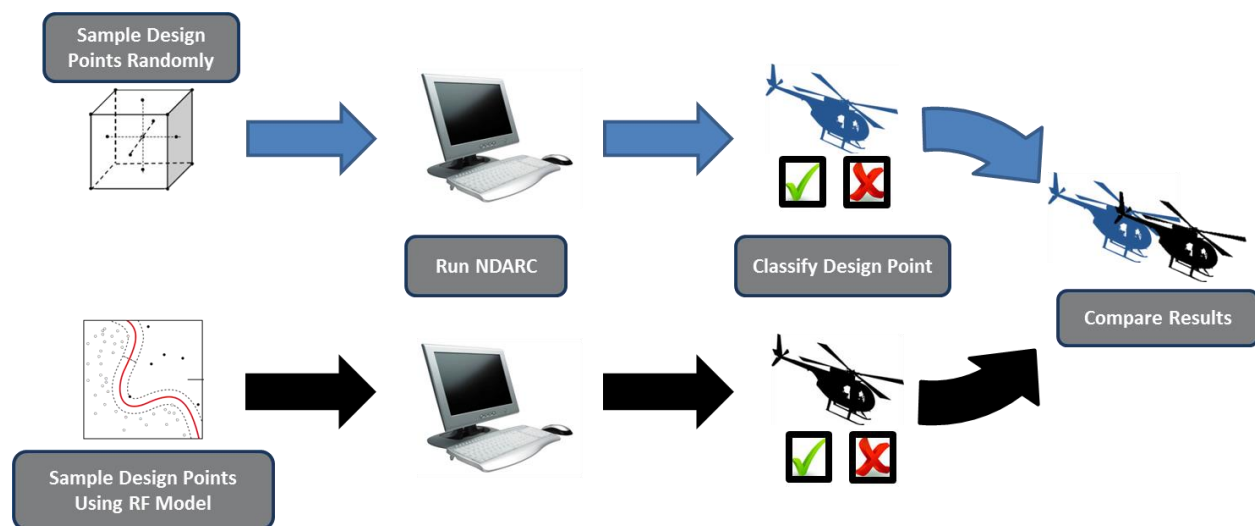
Figure 2-26: Approach used to test the proposed hypothesis and determine if SeBBAS approach yields significant results

Design Space

Five NDARC variables were varied from the baseline values during this case study. These five variable, along with their ranges are provided in Table 2-6. These five variables were randomly selected from the list of available NDARC variable in Appendix B. The study was limited to five design variables based solely on time constraints, and can be extended to any number of NDARC variables in the future.

Table 2-6: Design variables and ranges used during the NDARC case study

NDARC Design Variable	Lower Limit	Upper Limit
Diskload	5	10
CWs	0.04	0.15
TECH_blade	0.75	1.25
TECH_eng	0.75	1.25
Peng	1000	1800

Design Point Classification

No constraints were applied to the UH-60 performance results. Classification was made solely based on whether NDARC successfully converged (feasible) or failed to converge (infeasible) at a given design point. This was intentionally done to test the ability of the SeBBAS approach to handle "black-box" engineering functions that have zones of non-convergence.

Fitting Random Forest Model

For this case study, the Random Forest model was fit using 5000 training data points, with 5000 design points allotted for refinement.

Results of UH-60 Case Study

The UH-60 case study was repeated three times using the design variables outlined in Table 2-6, with the results shown below in Table 2-7. Though the Random Forest classification model did show an improvement in sampling feasible design points over a purely random sampling method, the results were by no means impressive. However, based on the validation data set the RF classification model was accurate 97.35% of the time. This would strongly indicate that the SeBBAS approach could produce significant results under the right circumstances. That is, one would expect the benefits gained from the SeBBAS approach to increase as the extent of the infeasible region of the design space increases.

Table 2-7: Results of UH-60 case study

Run	# Design Points	RF Model		Random Sampling	
		# Infeasible DP	% Infeasible	# Infeasible DP	% Infeasible
1	1000	22	2.20%	43	4.30%
2	1000	26	2.60%	61	6.10%
3	1000	30	3.00%	55	5.50%

Examining Table 2-7, the random sampling performed in this case study only found approximately 5% of the randomly selected sample space to be infeasible. To test the theory stated above, the same study was run again, with additional constraints placed on performance metrics of the helicopter in an attempt to create larger infeasible regions within the design space. Two arbitrary constraints were applied purely to restrict the feasible region of the design space:

1) Empty Weight < 11,000 lbs.
2) Total Helicopter Cost < $13 million

With these new constraints applied, approximately 46% of the design space was found to be infeasible based on the random sampling results. From the validation data set, the accuracy of RF classification model was about 97.56%, which was practically identical to the unconstrained case study. With this newly constrained design space, there is clearly an advantage gained from using a RF classification model to test design points for feasibility.

Table 2-8: Results from the constrained design space analysis

Run	# Design Points	RF Model		Random Sampling	
		# Infeasible DP	% Infeasible	# Infeasible DP	% Infeasible
1	1000	31	3.10%	445	44.50%
2	1000	37	3.70%	451	45.10%
3	1000	27	2.70%	484	48.40%

Two initial conclusions were drawn from this case study. First, as one might expect the RF classification model is more beneficial for problems that have large infeasible regions within the design space. This is simply because as the infeasible regions grow in size, a random sampling or structured DOE approach has a greater probability of sampling a design point from the infeasible region, whereas the RF classification model has at least some knowledge of the design space, allowing it to classify design points with relatively high accuracies. Secondly, even for design problems that aren't highly constrained, the RF classification

model still offers an improvement, though the computational effort required to fit the RF model may not be worth it in these cases.

2.7   Suggested Use of SeBBAS Approach

Once the SeBBAS approach has been used to fit a Random Forest classification model to the design space, it is up the user on how to use it. The approach suggested by this author is outlined in Figure 2-27. This approach takes advantage of the fact that a RF model fit in R provides not only a classification, but also the probability that the classification is correct. In actuality, for a given design point the RF model will calculate the probability that it should be classified as feasible (value of 1) or infeasible (value of -1), and it will return the classification that had the highest probability of being correct. For example, for a design point $\bar{x}$ the RF model might return $P(1) = 0.95$ and $P(-1) = 0.05$, the RF classification model would classify $\bar{x}$ as feasible. In this case there is a strong confidence that the design point is feasible, and thus the surrogate model would be used to evaluate the design point. On the other hand, if $P(1) = 0.51$ and $P(-1) = 0.49$, the RF model would still classify the design point as feasible. However, there is clearly little confidence in this classification being correct, and thus it would be prudent to evaluate this design point using the actual engineering model.

To perform this refinement, a threshold confidence limit, ε, must be selected. The threshold limit is simply the probability that the RF model classification is correct. Any design point for which the confidence level is above the threshold limit would be evaluated using the surrogate model, while those design points that have a high probability of being classified incorrectly according to the RF classification model are evaluated using the actual engineering model. This process attempts to correct for false classifications provided by the RF model, while minimizing the number of design points that have to be evaluated using the (potentially expensive) engineering model(s).

41
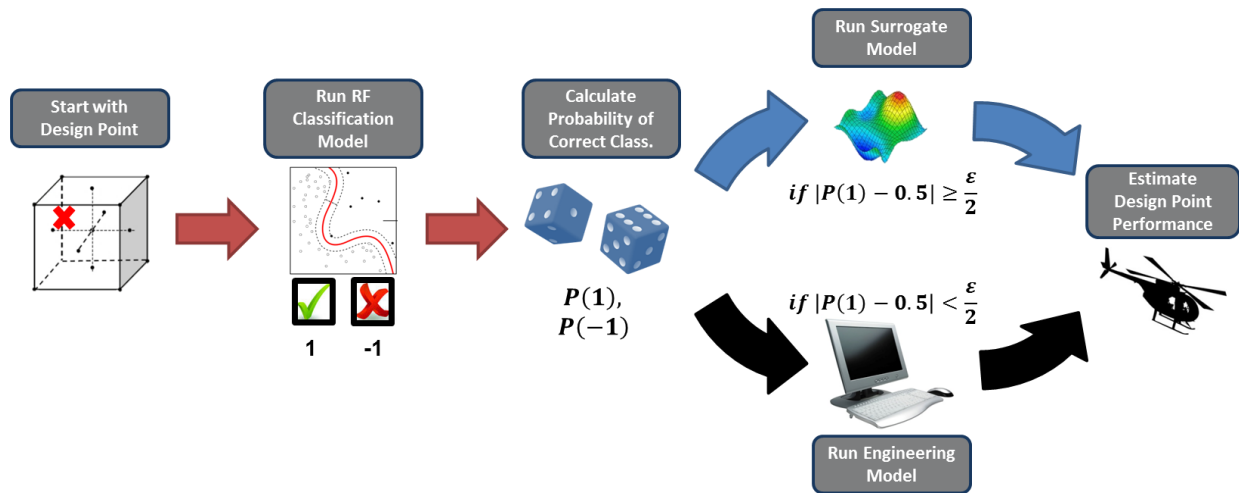
Figure 2-27: Suggested use of SeBBAS approach to determine whether or not surrogate model or engineering model will be analyzed based on the confidence of the RF classification

The process outlined in Figure 2-27 was performed using the case study presented in Table 2-8. A threshold confidence level of 80% was used in this study. The results, shown in Table 2-9, indicated that this method shows promise in identifying and correcting for false classifications provided by the RF classification model. In addition, depending on the availability of computational resources and time, the accuracy of this method can be improved by setting a higher threshold confidence limit requirement, at the cost of more calls to the engineering model. An example of this is shown by the results presented in Table 2-10, which used a threshold confidence limit of 90%. Here, slightly better accuracy was gained at the cost of approximately three times the computational expense, indicating diminishing returns on the accuracy gained as the threshold confidence limit approaches 100%.

Table 2-9: Study conducted to attempt to identify incorrect classifications from RF model using $\varepsilon = 0.80$

Run	# Design Points	Original % Correct Classifications	Refined % Correct Classifications	Number of Engineering Model Runs Required
1	1000	96.90%	98.00%	77
2	1000	96.30%	97.60%	107
3	1000	97.30%	98.30%	78

Table 2-10: Study conducted to attempt to identify incorrect classifications from RF model using ε = 0.90

Run	# Design Points	Original % Correct Classifications	Refined % Correct Classifications	Number of Engineering Model Runs Required
1	1000	96.90%	98.70%	251
2	1000	96.30%	97.90%	260
3	1000	97.30%	99.00%	243

2.8   Future Work

Based on work done by Justin Kizer in his PhD dissertation [2], the SeBBAS algorithm has been implemented using a combination of MATLAB and the R statistical programming language. This combination is required as the desired Random Forest machine learning algorithm is not available directly in MATLAB. However, the same Random Forest algorithm available in R is also available in Python. Thus, Python would provide a single, open source platform to implement the SeBBAS algorithm, which would streamline the process immensely. For example, as disused in Section 2.4 MATLAB and R communicate through .csv files that must be written and read during each iteration. Though this process is not prohibitively expensive, it does provide the opportunity for user errors if the .csv files are not located in the proper location. Because of this, it is highly suggested that the SeBBAS algorithm be transitioned from MATLAB/R implementation to a single Python based code.

Work must also continue on testing the SeBBAS algorithm using NDARC. As outlined in Section 2.6, a MATLAB function has been created to run NDARC and classify a design point as either feasible or infeasible. This function provides some flexibility on the selection of design variables, though more can be added if needed. To date, SeBBAS tests on NDARC have been successfully run with up to five design variables, resulting in a classification accuracy of approximately 97% when checked against validation data sets. Future work should focus on expanding the number of design variables from five to a sample size representative of CATEs modeling capabilities. The only limitation to selecting a larger subset of design variables is the computational expense associated with exploring a larger design space.

Caution should be used when setting the ranges for the design variables though. If the range of a design variable is too large (thus the design variable takes on infeasible values), it may cause NDARC to enter an infinite loop and never converge. If the design variable ranges are found to be reasonable and the code still enters an infinite loop (i.e. NDARC never completes its run and return command to SeBBAS), then it is likely that NDARC is having issues with the solvers used within the code. If this is the case, further

investigation is required to identify the underlying cause of the issue before the SeBBAS algorithm can successfully be run. [2]

3.   Create a method for selecting and applying multiple quantitative technology forecasting techniques to a specific rotorcraft configuration

3.1   Introduction

Within the engineering community as a whole, technology forecasting involves assessing the impact of emerging technologies on future system design and performance. [5] Complex system design relies heavily on accurate forecasting as decision makers utilize these forecasts to better understand the problem at hand. Technology forecasting can be a difficult endeavor as there are many uncertainties associated with the process. As such, properly selecting and implementing a technique, or set of techniques, is vital to future system design. This study seeks to develop a methodology to select the proper technology forecasting technique, or set of techniques. First the motivation of such effort is discussed, followed by an extensive literature search that must be performed in order to understand existing technology forecasting techniques. Next, a methodology is developed to aid selecting the appropriate technique for the given study. Finally, the methodology implemented in the form of a Microsoft Excel-based decision support tool and demonstrated with a technology applicable to complex systems of interest, particularly to Rotorcraft.

3.2   Motivation

Previous research with the Capability Assessment and Tradeoff Environment (CATE) has identified the need for an improved technology forecasting approach. [6] CATE currently utilizes a "k-factor" approach to estimate performance impacts. This approach utilizes quantitative representations of technologies by estimating their impacts as changes to baseline metrics. In order to develop these estimated impacts in the past, the Del-phi method was used. The Del-phi method is essentially obtaining expected technology impacts from expert elicitation. There are several draw backs to this type of approach. First, the process of locating and contacting Subject Matter Experts (SMEs) can be difficult and time consuming. Second, educating the SMEs on the method itself can also be strenuous task. Finally, the results of iterating with the SMEs can be prone to bias as they want to "sell" their technology. This can lead to poor technology forecasts. Understanding these difficulties has led to the need for a better way to forecast a given technology. There is a need to implement new technology forecasting techniques to better represent technologies within the CATE environment.

3.3   Background and Literature Search

There are two main ways to characterize a given technology forecasting technique: (1) normative and (2) exploratory. Normative technology forecasting involves determining a course of action to help reach a future goal. On the other hand, exploratory technology forecasting is used when a decision maker desires to predict the future state of a given technology area. [7] There are many different types of forecasting

techniques, but it is important to note that they can be fit into 9 basic families of techniques: Expert Opinion, Trend Analysis, Monitoring and Intelligence methods, Statistical Methods, Modeling and Simulation, Scenarios, Valuing/Decision/Economics Methods, Descriptive and Matrices Methods, and Creativity. [8]

Upon understanding the motivation and background for the task, it is important to perform an extensive literature search in order to better understand the task at hand. This literature search consists of three main components: (1) understanding the families of forecasting techniques in some detail, (2) understanding what forecasting techniques are related to the complex systems of interest, and (3) understanding the current state of technology forecasting technique selection.

3.3.1   Families of Forecasting Techniques

Each of the families of forecasting techniques utilizes a different approach to forecast a given technology based on the characteristics of the technology itself as well as the desired outcome of the forecast. The following descriptions are based on the work of Firat, unless otherwise noted. [9]

Expert Opinion

Methods in the Expert Opinion family understand or forecast technological development utilizing intensive discussions with subject matter experts. The most common method in this family is the Del-phi Method, which is discussed in the previous section. Firat states that this method "combines expert opinions concerning the likelihood of realizing the proposed technology as well as expert opinions concerning the expected development time into a single position." Essentially, iterating with the SMEs will lead to an expected system-level impact of a given technology while maintaining an understanding of the time it will take to realize that technology.

Trend Analysis

Trend Analysis methods involve prediction of a technology's impact utilizing quantitative historical information and continuing it into the future. Such methods include economic forecasting models and techniques such as regression, exponential smoothing and Box-Jenkins' ARINA model and growth curve fitting. [10] These methods utilize the fact that a technology usually has a life cycle of development that follows some kind of trend. For example, growth curve fitting utilizes an estimation of a technology's life cycle development curve. This curve is then used to forecast the technology's impact into the future based on its development up to its current state

Monitoring and Intelligence Methods

Monitoring and Intelligence methods are suitable for making one aware of changes on the horizon that could impact the penetration or acceptance of the technologies in the marketplace. [11] Resource availability is an issue for these methods given the fact that many of these methods require "scanning a technology's environment" in order to understand the impact of adoption. This means that experts must be identified and "tracked" by maintaining contact with them.

Statistical Methods

Two of the most popular methods in the Statistical Methods family are Correlation Analysis and Bibliometric Analysis. Correlation Analysis forecasts the development of a new technology when the development patterns of an existing, related technology are known. Martino describes one such method, i.e. Lead Lag Correlation, by comparing the time lag between the development of composite components in aircraft and the application of such components into actual aircraft as shown in Figure 3-1 Bibliometric Analysis involves a text mining approach to search existing publications as well as existing and up-and-coming patents. [12] An important aspect of bibliometric analysis is that it goes beyond expert biases by using sound data from published results.
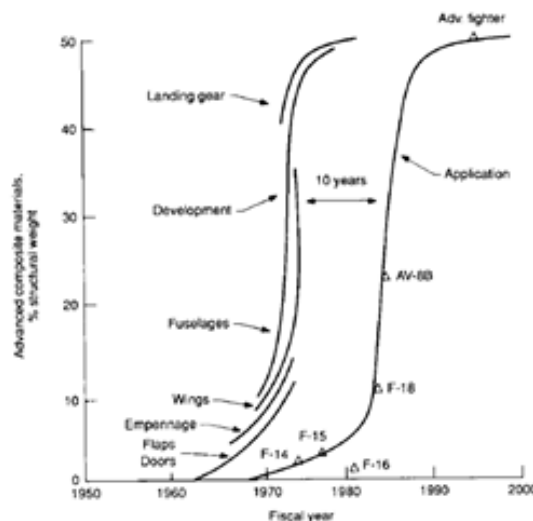
Figure 3-1: Time Lag from Development to Application of Advanced Composites in Aircraft

Modeling and Simulation (M&S)

Modeling and Simulation methods utilize the development of a "model" of the "real world" where a technology of interest can be infused in order to understand its impacts at a system level. One such method is a Causal Model. The development of causal models requires the understanding of what causes technological change. Martino introduces what's called Technology-Only causal models that assume "technological change can be fully explained by factors internal to the technology-producing system." [12]

In his discussion of this type of causal model, Martino discusses the universal growth curve developed by Floyd (1968) which attempts to explain growth toward an upper limit based on the effort extended by active researchers. [13]

Scenarios

The Scenarios family of methods seeks propose different concepts of future technology based on a well-defined set of assumptions. [9] Each different concept, or alternative, represents different characteristics of a future technology. Each concept is then evaluated against the assumptions and results in being able to determine the scenario most likely to occur. [10]

Valuing/Decision/Economics Methods

According to Levary and Han, the most popular method in this category is the "relevance tree approach". This normative approach to technology forecasting involves breaking down the goals and objectives of the technology of interest into lower level goals and objectives in a hierarchical format. The probabilities of achieving each goal and objective in the various levels of the "tree" must be estimated, allowing the decision maker to forecast the likelihood of achieving the stated goals and objectives of the technology of interest. [10]

Descriptive and Matrices Methods

There are two main methods in this family of techniques: technology road-mapping and analogies. Technology road-mapping consists of projecting major technological elements of product design and manufacturing together with strategies for reaching desirable milestones efficiently. One such forecast related to the complex systems of interest to CATE are the goals set forth in the Aviation Science & Technology Strategic Plan (ASSP). [14] Such roadmaps aid decision makers by providing a vision for where a given technology area may be going in the future. Analogies involve a systematic comparison of an up-and-coming technology with an existing technology that is believed to have been similar in some respects. [9] The down side of analogies is that there is no guarantee that technologies being developed today and in the future will in fact follow the same development process as past technologies. As such, these forecasts are probable at best, but never certain. [12]

Creativity

The creativity family of methods is the most interesting. It includes methods such as brainstorming and science fiction analysis. Brainstorming involves simply thinking about where a technology may go using sound engineering judgement, while science fiction analysis involves looking at technologies used in science fiction novels and movies and trying to understand the author's or writer's basis for conjuring up

such technology. Due to the creative nature of such methods, techniques in this category can only give a decision maker a direction that a technology may go and not any concrete results.

3.3.2 Forecasting Techniques Related to Complex Systems

Given the wealth of information on forecasting techniques, it is important to understand the techniques that are related to the complex system design of interest. Previous research as part of the CATE development effort, involved developing a taxonomy of forecasting techniques for complex systems. [15] In his paper, Smith compiled a taxonomy of forecasting techniques related to complex system design utilizing a text mining approach. The development of the taxonomy involved 3 steps: (1) techniques were compiled based on results of existing literature surveys, (2) the techniques were screened utilizing the text mining approach, and (3) the techniques were characterized based on criteria relevant to complex systems. As a result of this effort, a taxonomy of 60 techniques was compiled, utilizing techniques across all 9 families of techniques. After performing an extensive literature search, Smith developed a way to describe techniques which resulted in 12 characteristics found in Table 3-1 below. [16] [17] [18]

Table 3-1. Technology Forecasting Technique Characteristics

Technique Characteristics

1. Capability to forecast incremental change
2. Capability to forecast radical innovations
3. Capability to forecast modular technologies
4. Life cycle prediction capability
5. Capability to forecast for stipulated time horizon
6. Data availability
7. Data validity
8. Technology development predictability
9. Technology similarity
10. Method of adaptability
11. Ease of technique implementation
12. Cost of technique implementation

More details about the approach used to compile this list of characteristics, as well as a description of each technique characteristic from Table 3-1, can be found in Intepe's work. [16]

3.3.3   Current State of Technology Forecasting Technique Selection

Given the number of forecasting techniques that exist, there is a need to aid decision makers in selecting the appropriate technique to use for their analysis. One such approach is described by Mishra and Deshmukh and is outlined herein. [18] Mishra discusses the steps for selecting an individual forecasting technique. It begins with rating the technology of interest based on characteristics that are related to the characteristics of the techniques. One such mapping is shown in Table 3-2 below. Next a multi-criteria decision making technique is used to evaluate the techniques and select the technique that is closest to the ideal technique. One such MCDM technique that could be used is Technique for Order Preference by Similarity to Ideal Solution (TOPSIS).

Table 3-2. Matching Technology Characteristics to Technique Characteristics

Technology Characteristic	Technique Characteristic
Evolutionary change	Capability to forecast incremental change
Revolutionary change	Capability to forecast radical innovations
Modularity of technology	Capability to forecast modular technologies
Life cycle	Life cycle prediction capability
Time frame of interest	Capability to forecast for stipulated time horizon
Existing data availability	Data availability
Exiting data validity	Data validity
Technology readiness level	Technology development predictability
Existing similar technologies	Technology similarity
Amount of existing information	Method of adaptability
Time available for study	Ease of technique implementation
Resources available for study	Cost of technique implementation

The method described above can be utilized to select a single forecasting technique that can be applied to the desired technology to be forecasted. Given the complexity of the systems to which technologies can be infused, accurate forecasts are vital. As such, developing a way to select multiple techniques can provide a more accurate forecast.

3.4   Methodology and Implementation

The overall approach used for this effort is outlined in Figure 3-2. The first step involves developing and refining a methodology, applicable to CATE, to select technology forecasting techniques. This involves leveraging the taxonomy of forecasting techniques developed by Smith. [15] Doing so would limit the scope of the problem, with the focus now being on developing a selection methodology rather than further

exploring the forecasting techniques themselves. The next step is to demonstrate the developed methodology on a technology related to the UH-60 Blackhawk. This is desired as the UH-60 is used as a case study under the CATE research efforts. The UH-60 is a relevant system that is still being used today as well as looking to be upgraded for future endeavors. Finally, integrating this methodology into the CATE environment would allow decision makers to get more accurate forecasts of technologies they desire to infuse onto the complex system of interest

Figure 3-2. Outlined Approach

3.4.1 Methodology Development

Most of the effort for this research effort is concentrated on the first step of the overall process in Figure 2 due to time constraints. Figure 3 illustrates a closer look at this first step.
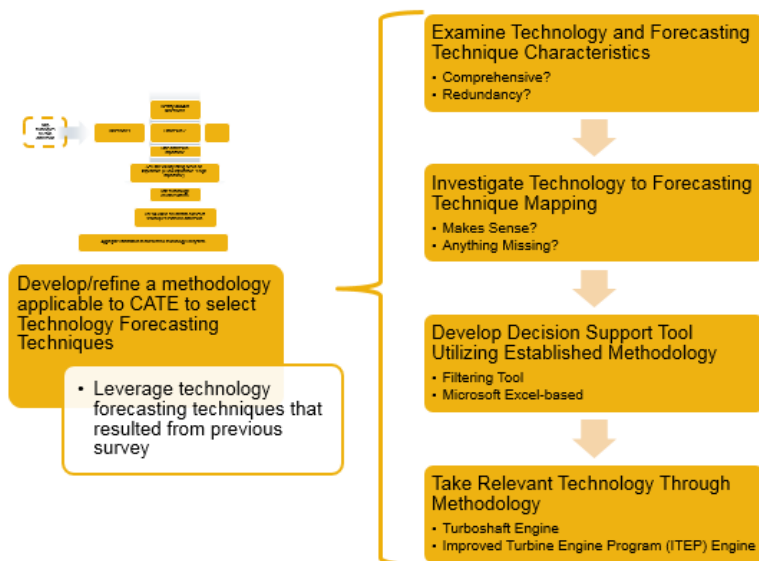
Figure 3-3. A Closer Look at Step 1

As shown in Figure 3-3, this step begins by examining the technology and technique characteristics set forth by Smith and Mishra what are contained in Table 3-2. [15] [18] This examination involves

51

determining if this list is not only comprehensive but also if there are any redundancies as well. Next, the mapping from technology characteristics to forecasting technique characteristics is explored. There needs to be a way to select a technique based on the technology's characteristics. After investigating the mapping, a Microsoft Excel-based decision support tool is developed in order to implement the methodology in a usable fashion. Finally, this process is demonstrated utilizing a relevant technology. Given most rotorcraft are powered by turboshaft engines and upgrades are always being performed to improve engine performance, a turboshaft engine upgrade is take through this process, namely the 3,000 shaft horsepower Improved Turbine Engine Program (ITEP) engine. Each sub-step in Figure 3-3 is explained in detail below.

Examine Technology and Forecasting Technique Characteristics

For the purpose of this research, the list of technology and technique characteristics is inclusive enough as well as without any redundancies. This list is compiled utilizing the works of Mishra, Cheng, and Intepe. Mishra validated the first 5 characteristics in Table 3-2 through a questionnaire given to 45 random technology forecasting experts and he selected characteristics encompass many aspects of a given technique [18]. Characteristics 6-12 are used by both Cheng and Intepe in separate efforts. [17] [16]

Investigate Technology to Forecasting Technique Mapping for Selection

With the characteristics of both the technology and techniques now established, the next step is to begin developing a way to select a technique, or subset of techniques, to be used for forecasting. Initial thoughts were to just utilize the TOPSIS method described by Mishra. [18] The TOPSIS process is done with the technology's characteristics being scored and used as the ideal solution. Then the characteristics of each technique are evaluated and the techniques are ranked based on their relative distance to the ideal technique (i.e. how close they are to the technology characteristics).

During the implementation of this method, it was discovered that this is not very effective given the taxonomy developed by Smith that is being used. In order to use Mishra's method, each technique's characteristic needs to be scored on a scale of 1-10, with 1 meaning that the technique does a poor job with the characteristic and 10 meaning that the technique performs the characteristic exceptionally. The taxonomy developed by Smith scores each characteristic with either a 0 or a 1, with 0 meaning it does not have this characteristic and 1 meaning it does. Utilizing this method resulted in the technique with the most 1's for its characteristics being selected as the best technique. This makes sense because if a technique has every characteristic, it can perform a forecast for almost any technology. This is undesirable as a technique can be over performing beyond what is necessary. More potential mappings needed to be explored.

Intepe and Cheng utilized similar approaches but using fuzzy logic. [16] [17] After exploring their results, the same conclusions can be made. Given the fact that the taxonomy only scores each characteristic with a

0 or a 1 rather than on a scale, it is ineffective to use any of the multiplicative approaches. There is a need for a new approach in order to utilize the taxonomy developed by Smith. It is important to note that the scores for each characteristic in Smith's taxonomy could be reevaluated in order to score them on a scale of 1-10. However, given the scope and timeframe of this research, developing the questionnaires and requesting experts to rate each technique characteristic was not possible. Future work in this area should consider surveying experts. For this period of work, a method was developed to filter the technology forecasting methods that could be utilized that have a total commonality score above a given threshold when compared to the technology characteristic scores. This method does not allow a decision maker to select a technique, but it reduces the number of techniques that need to be explored in greater detail, saving time and resources. Essentially, each characteristic of the technology of interest is evaluated with whether or not each technique can perform an analysis based on that characteristic. For example, if a technology is an evolutionary technology, such as an engine upgrade, the technique to be used will need to be able to forecast an evolutionary technology. If it can, it is given a score of 1; if it cannot, it is given a score of 0; and this is applied to all characteristics. Some of the technique characteristics are given a score of 0.5. This indicates that the technique can "somewhat" handle that type of characteristics. Although this method is not an exact science, it does provide a qualitative way to handle such characteristics.

Each technique characteristic is weighted in order to determine the relative importance of each characteristic. This is important as it means that the more important characteristics achieve a higher score based on their higher importance to the forecast, while characteristics of lower importance receive a lower score. The weightings of each characteristic are taken from Mishra. Mishra developed a questionnaire that was then given to technology experts. The experts came to a consensus on what Mishra calls the "inter characteristic weightage" (i.e. the weightings of each technology forecasting technique characteristic). [18] The technique ratings from Mishra are shown in Table 3-3.

Table 3-3. Technique Characteristics Weighting

Technique Characteristic	Weighting
Capability to forecast incremental change	10
Capability to forecast radical innovations	10
Capability to forecast modular technologies	5
Life cycle prediction capability	8
Capability to forecast for stipulated time horizon	5
Data availability	10
Data validity	8
Technology development predictability	10
Technology similarity	8
Method of adaptability	8
Ease of technique implementation	10
Cost of technique implementation	10
Quantitative or Qualitative	10
Exploratory or Normative	10

Next, the characteristics of the technology and the characteristics of each technique are compared for commonality. This is done in a logical progression of four steps.

If the technology characteristic score is the same as the technique score, this is considered *highly favorable* and given a comparison score of 1.

1. If the technique characteristic score is greater than the technology characteristic score, this is also considered *highly favorable* and given a comparison score of 1.

2. If the technique characteristic score is 0.5 and the technology characteristic score is 1, this is considered *favorable* and given a comparison score of 0.5.

3. If the technology characteristic score is a 1 and the technique characteristic score is a 0, this is considered *unfavorable* and given a comparison score of 0.

Finally, each of the commonality scores for each technique are summed to obtain a final total technique commonality score. This technique commonality score is a measure of how common a technique's characteristic scores are to the technology's characteristic scores. The techniques with comparison scores that are above a user-defined threshold are selected to be filtered for further exploration in greater detail.

Develop Decision Support Tool Utilizing Established Methodology

The final step is to implement this methodology into a Microsoft Excel-based decision support tool that allows the user to answer questions about the technology he or she would like to forecast. This results in a list of techniques that can be further explored in more detail, with the end result being the ability to select a forecasting technique. Figure 3-4 below outlines the data flow of the tool. The images in this sub-step are snapshots of the tool itself.

The first step in tool's data flow (outlined in Figure 3-4) is a technology questionnaire. The user is asked 14 questions about the technology as shown in Figure 3-5. 11 of the questions are related to the 12 characteristics of the technology from Table 3-1. One question asks whether a quantitative or qualitative approach is desired and one question asks whether a normative or exploratory approach is desired. These questions are used in conjunction with the classifications of each technique as either quantitative or qualitative and exploratory or normative and help select the most desirable techniques. The final question is about an acceptable score threshold to filter out a technique for further exploration. This question is answered with a slider bar. For questions 2-8, an error message is displayed if both or neither check boxes are selected as only one can be utilized for a successful filtering effort. Once scored, the user will click the "Filter Techniques" button which resides below the Technology Questionnaire in order to obtain a filtered list of acceptable techniques to be further explored in greater detail.

Figure 3-4. Tool Data Flow

Figure 3-5. Technology Questionnaire

Once the "Filter Techniques" button is clicked, each technology characteristic is scored based on the answers to the Technology Questionnaire. The characteristic can either be give a score of 0 or 1 for each of

55

the first 11 questions, while certain text is used for the answers to questions 12 and 13 (depending on the answer) as shown in blue column in Figure 3-6. These are based on how the techniques are categorized in the taxonomy. [15] The answer to question 14 is used to filter the forecasting techniques later in the process.

Technology

Characteristic	Score
Evolutionary change	1
Revolutionary change	0
Modularity of technology	1
Life cycle	0
Time frame of interest	0
Existing data availability	1
Existing data validity	1
Technology readiness level	0
Existing similar technologies	1
Amount of existing information	0
Time available for study	1
Resources available for study	0
Quant (H) or Qual (S)	H
Exploratory or Normative	Ex

Results from "Questionaire"

Figure 3-6. Technology Characteristic Scores

Next, and most importantly, the technology characteristic scores are compared to the technique characteristic scores using the logic described in sub-step 2 of this process. Each characteristic score is multiplied by a weighting factor based on how important that characteristic is to the technology forecasting technique selection process as also described earlier. This results in a "vector" of commonality scores for each technique in the taxonomy comparing its characteristics to the technology desired to be forecasted. Next each "vector" of commonality scores is summed to obtain a total commonality score for each technique.

Finally, a list of acceptable forecasting techniques is displayed to the user. The acceptable techniques are determined based on the threshold score from question 14 of the technology questionnaire. The techniques with a total commonality score greater than this threshold are displayed in an alphabetical list, as shown in Figure 3-7, to the right of the technology questionnaire. As can be seen in Figure 3-7, there is also a bar chart that illustrates the distribution of the total commonality scores for all of the techniques. This will allow the user to alter the acceptable score threshold (question 14 of the technology questionnaire) in order to better filter the techniques based on the user's purpose. Once this list is obtained, the user can now explore a sub-set of the technique taxonomy that contains techniques that are more applicable to the technology of interest as well as the user's own forecasting preferences.
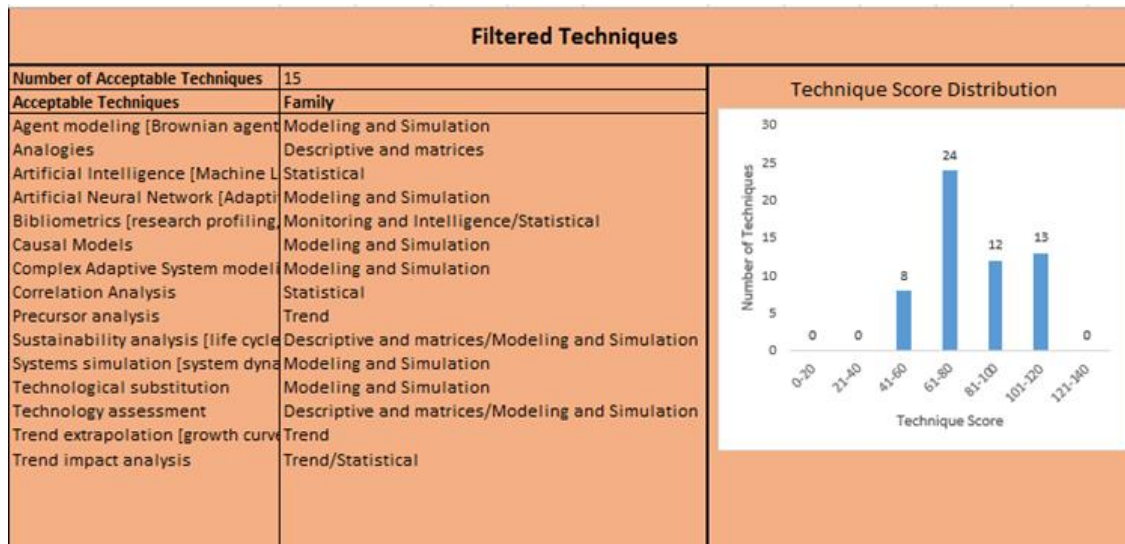
Figure 3-7. Filtered List of Acceptable Forecasting Techniques

Take Relevant Technology through the Methodology

The final step is to demonstrate this process with a relevant technology. For this demonstration, the 3,000 shp Improved Turbine Engine Program (ITEP) engine was selected as it is in development by General Electric (GE), now designated as the GE T901, as it is very relevant to the UH-60 Blackhawk upgrades. GE is expected to begin production of the engine in 2025. It is designed to produce 50% more power at SL/ISA, reduce fuel consumption by 25%, and have 20% longer life than compared to the GE T700 family of engines currently being used on the UH-60. [19] For this demonstration, it is important to note that the purpose of such forecast is to understand how the power-to-weight ratio (shp/lb) of the engine will be improved at the time of the engine's production, as this influences the vehicles vertical rate of climb (VROC) and maximum forward speed, both of which are of interest to UH-60 operators today. There are many other performance parameters that could be explored, such as specific fuel consumption and efficiency, but the power-to-weight ratio is the most applicable at this time.

To begin the process, information is needed about the prospects of the engine and the desired forecast in order to complete the Technology Questionnaire. As most engines, the ITEP engine is an evolutionary technology as it is based on previous engines built by GE. The engine is expected to operate with current UH-60 aircraft without a redesign, indicating that it is modular in nature (i.e. can be swapped with current engines). For the purposes of simply determining power-to-weight ratio, the ability to forecast the lifecycle of the engine is not necessary. Given the engine is expected to begin production is 2025 (less than 10 years), the timeframe for the engine's adoptions is assumed to be "short-term" as it is already in development, though in its early stages. Given that many turboshaft engines are already in use and the turboshaft engine

development began over 60 years ago, it is assumed that there is a lot of data that can be obtained and that this data is considered reliable. This fact also means that there are many similar technologies. For this forecast, it is desired that expert opinion is not utilized, as this is the goal of this task for the CATE research team. Given that the outcome of the forecast will be used in a Modeling and Simulation environment, such as the CATE environment, a quantitative and exploratory approach needs to be used. A total commonality score of 100 will be used as an acceptable score to keep a technique for further exploration. Utilizing this information, the Technology Questionnaire is answered as shown in Table 3-4 below.

Table 3-4. ITEP Engine Demonstration Technology Questionnaire Answers

Question				
1. Is this technology evolutionary or revolutionary?	x	Evolutionary		Revolutionary
2. Is the technology modular in nature (or having modular components)?		No	x	Yes
3. Would you like to forecast the life cycle of the technology?	x	No		Yes
4. What is the timeframe for the technology's adoption?	x	Short-term		Long-Term
5. How much data do you have access to?		A little	x	A lot
6. How reliable is this data?		Not Reliable	x	Reliable
7. Where is the technology in its development?	x	Early Stages		Late Stages
8. How many similar technologies exist?		A little	x	A lot
9. Will you need to rely on expert opinion?	x	No		Yes
10. How much time do you have to spend on forecasting?		A little	x	A lot
11. What are your resources like?	x	Minimal		Many
12. Would you like a quantitative or qualitative approach?*	x	Quantitative		Qualitative
13. Would you like a normative or exploratory approach?*		Normative	x	Exploratory
14. What is an acceptable score to keep a technique?		100		

With these answers to the Technology Questionnaire, the "Filter Techniques" button is clicked and 15 forecasting techniques are filtered for further exploration, shown in Table 3-5, all with a total commonality score greater than the indicated 100 threshold. The forecasting technique family is also recovered from the taxonomy and displayed.

Table 3-5. ITEP Engine Demonstration Filtered Forecasting Techniques

Filtered Techniques	Family
Agent modeling [Brownian agents]	M&S
Analogies	Descriptive and matrices
Artificial Intelligence [Machine Learning]	Statistical
Artificial Neural Network [Adaptive neuro-fuzzy inference]	M&S
Bibliometrics [research profiling, patent analysis, text mining, citation network analysis]	Monitoring & Intelligence/Statistical
Causal Models	M&S
Complex Adaptive System modeling (CAS) [Chaos]	M&S
Correlation Analysis	Statistical
Precursor analysis	Trend
Sustainability analysis [life cycle analysis]	Descriptive and matrices/M&S
Systems simulation [system dynamics, KSIM]	M&S
Technological substitution	M&S
Technology assessment	Descriptive and matrices/ M&S
Trend extrapolation [growth curve fitting and projection]	Trend
Trend impact analysis	Trend/Statistical

In order to examine if the selected threshold score of 100 is acceptable, it is important to look at the distribution of commonality scores. This distribution is shown in Figure 3-8. Examining Figure 3-8, it is clear that a total commonality score of 100 is a good lower bound to filter out the acceptable techniques as all of the techniques have a total commonality score less than 120, and 15 techniques is a reasonable number of techniques to further explore without too much effort.
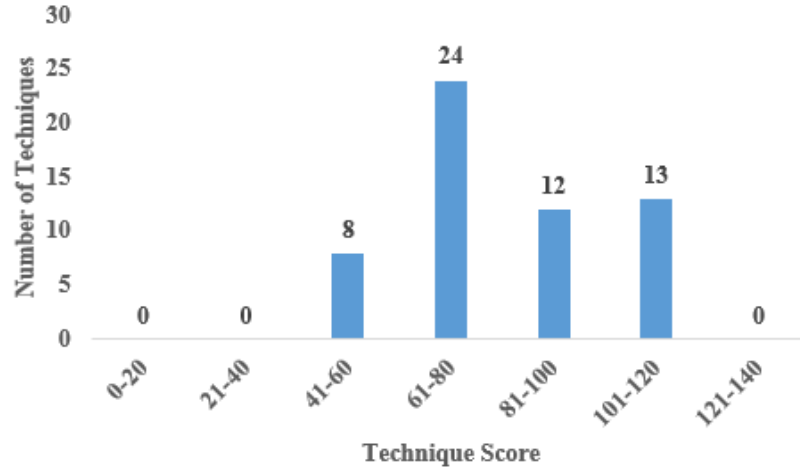
Figure 3-8. Technique Commonality Score Distribution

The next step is to explore the filtered techniques. Due to time constraints of this research effort, the two highest-scoring techniques were selected for further exploration: Causal Models and Trend Extrapolation. Causal models are in the Modeling and Simulation technique family and are exploratory in nature. There are two categories of causal models. The first are closed-form, analytical models (equation or set of equations) and the second are simulation models (set of differential equations). In addition to these two categories, there are three types of causal models: technology-only models, techno-economic models, and economic and social models. [12] For the purpose of this exercise, the technology-only model is used as it is the most applicable.

The Universal Growth Curve, as discussed by Martino, is a regression equation used to model the progression of a functional capability (f) of a technology that approaches an upper limit (F). [12] It also utilizes the functional capability of a competitive technology (f_c). [12] The regression equation is as follows:

$$\ln(Y - 1) + Y = C_t$$

(1)

$$\text{where } Y = \frac{F - f_c}{F - f}$$

It is important to note that C_t is not the coefficient of thrust; it is simply a constant parameter determined from the data points. It is assumed that the power-to-weight upper limit for turboshaft engines is 10, while the reciprocating engine is used as the competitive technology (as it was common when turboshaft engines were first being developed), with a power-to-weight ratio of 0.6 at the time. More details about this method can be found in Martino's paper.

60

Using the regression equation above, it is possible to construct an "s-curve" to model the progression of turboshaft engine power-to-weight ratio over time. This is done utilizing turboshaft engine power-to-weight ratios vs development year data points from Leishman's "Principles of Helicopter Aerodynamics" and are shown in Table 3-6 below. [20]

Using these data points, with each approximate power-to-weight ratio as a different f value, a C_t value can be calculated for each power-to-weight ratio in Table 3-6. Then, each C_t value is plotted against its corresponding year of development, with a linear regression equation is fit to these points. This regression is then used to find C_t values for various years from a time before the first data point in Table 3-6 to, and beyond, the year that the forecast is desired for. Then, using these C_t values corresponding Y values must be iteratively determined (based on the relationship between C_t and Y from Equation 1) and finally power-to-weight values can be determined for each Y value. Figure 3-9 illustrates the progression of turboshaft engine power to weight ratio over time from 1945 to 2050 with the regression equation also displayed. In order to utilize this forecast, one simply enters the year of a future turboshaft engine's development into the regression equation (i.e. x) and the result (i.e. y) is the power-to-weight ratio of that turboshaft engine.

The plotted regression equation is:
$$y = 2E\text{-}07x^4 - 0.0018x^3 + 5.4119x^2 - 7230.2x + 4E\text{+}06$$
$$R^2 = 0.9993$$
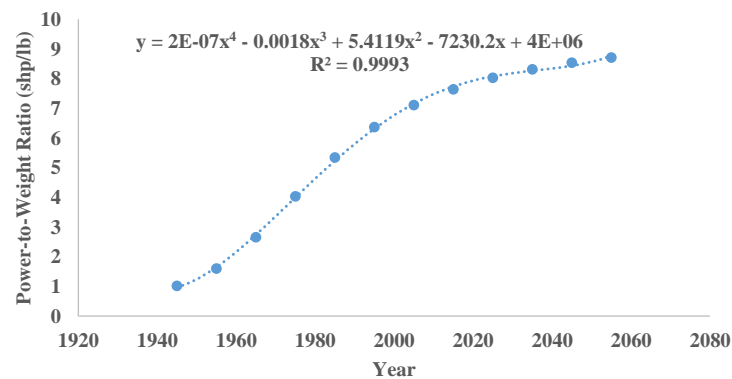
Figure 3-9. Turboshaft Engine Power-to-Weight Ratio Progression Utilizing the Technology-Only Causal Model

Table 3-6. Turboshaft Engine Power-to-Weight vs. Development Year Data Points

Year of Engine Development	Power-to-Weight Ratio (shp/lb)
1958	1.959
1979	4.149
1984	4.686
1987	6.302

Table 3-7 illustrates using this method for two different turboshaft engines: the GE T700, the first generation of the current UH-60 Blackhawk engine as well as the future ITEP engine. It is assumed that the ITEP engine has a development start year of 2016 as this is when GE was officially awarded the contract to begin developing the engine by the US Department of Defense (DoD). [21]

Table 3-7. Turboshaft Engine Forecast with Technology-Only Causal Model Results

Engine Name	Year of Development	Forecasted P/W (shp/lb)	Actual (Expected*) P/W (shp/lb)	Pct. Error (%)
GE T700	1976	4.129	4.15	0.50%
ITEP	2016	7.784	7.24	7.46%

From Table 3-7, it can be concluded that the Causal Model can predict a future turboshaft engine's power-to-weight ratio rather accurately but with some uncertainty. This uncertainty is mostly attributed to the sensitivity of the regression curve in Figure 3-9 to the upper limit (F) as well as the competitive technology value (f_c) used in the beginning of the model's development. There is also a level of uncertainty in the data points obtained from Leishman as they are taken from a graph of power-to-weight ratio vs year of development.

Next the Trend Exploration technique is considered. Trend Extrapolation is in the Trend technique family and is also Exploratory in nature. The Trend Extrapolation method utilized for this demonstration is the Exponential Trends method. There is also a Qualitative Trend method that is also part of this family, but in order to be applicable to the CATE research effort, Quantitative methods are desired. Overall, Trend

$$\ln(y) = Y = \ln(y_0) + kt \tag{2}$$

Extrapolation methods are simpler to implement when compared to the Causal Model. Using the same data points from Table 3-6, a curve fit is constructed by regressing the natural logarithm of the data points versus time utilizing the following regression equation: [12]

Using the data points in Table 3-6 as y values and Equation 2, it is possible to construct a graph of the engine power-to-weight ratio versus time as shown in Figure 3-10.
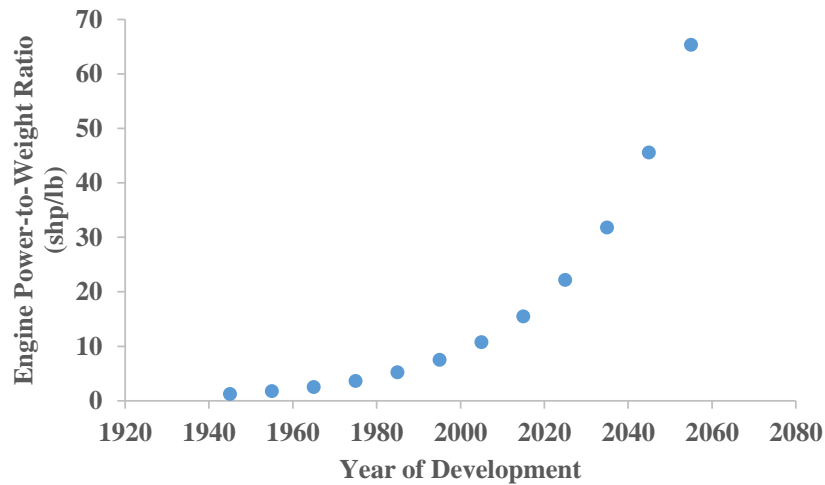
Figure 3-10. Turboshaft Engine Power-to-Weight Ratio Progression Utilizing the Trend Extrapolation Method

As can be seen from Figure 3-10, the trend, as the forecasting technique's name implies, follows an exponential trend. Upon further inspection of Figure 3-10, it is possible to see that by the year 2025, the expected year of production of the ITEP engine, turboshaft engines will have a power-to-weight ratio of about 20 with it exponentially increasing to over 30 almost ten years later. This is not an acceptable forecast for the power-to-weight ratio of turboshaft engines. These extremely large results are most likely due to the fact that the trend extrapolation method does not take into consideration the fact that fundamental limits need to be considered as a turboshaft engine power to weight ratio of about 65 is expected by the year 2055. This does not make sense as by then other technologies will be in play such as hybrid electric technologies and possibly even fuel cells as battery technology evolves.

This demonstration shows that the developed tool can successfully aid decision makers in selecting a sub-set of the technique taxonomy for further exploration. It can filter techniques that can be used effectively, such as the Causal Model. In potential use of this method, other high-scoring techniques should also be considered while evaluating the technology.

3.5   Conclusion

This study utilized an existing taxonomy of technology forecasting techniques and developed a methodology to select a sub-set of techniques from this taxonomy for further exploration. This methodology was implemented into a Microsoft-Excel based decision support tool that can help decision makers reduce the number of technology forecasting techniques to explore based on the forecasting task at hand. A demonstration was done using the turboshaft engine in the Improved Turbine Engine Program (ITEP) in

order to show the capability of the tool. This demonstration resulted in list of 15 techniques. Two of which were explored in more detail. The Causal Model was deemed applicable to the turboshaft engine upgrade forecast, while the Trend Extrapolation was not. This implies the methodology as well as the tool both need improvements to be more effective to the decision maker. These improvements can become future research tasks for the CATE research team. First and foremost, research is necessary to change the ratings of each technique's characteristics from discrete 1's and 0's to a scale of how well a technique handles that characteristic (rather than whether it can or not). This will allow the implementation of a decision-making technique to select the best forecasting technique (closest to the ideal technique) utilizing a method like TOPSIS. Another area of improvement is the link between the Technology Questionnaire in the tool and the technology score. Some of the questions in the questionnaire should be able to be answered with a slider bar rather than check boxes. For example, the question related to how much data is available can be answered with either "a little" or "a lot" when the amount of data available can vary between no data at all and plentiful data, not just the two end points. Adding some kind of scale will remove any ambiguity in the questionnaire. Another potential area of improvement is the commonality aspect of the tool. Currently there is a simple method to determine if a technique characteristic is common to the technology's characteristic, but it appears there need to be a more systematic way of determining this.

Though there are many improvements necessary to make the methodology and the tool more effective, this research results in a framework to build upon, and with a little more research can be used to quickly and efficiently select the appropriate forecasting technique to use in any research effort.

4. RCAS-CATE Optimization of RCAS Representation in NDARC Using Optimization Schemes

4.1 Introduction

The following section documents the work done in improving vehicle performance analysis methods including quantification of uncertainty. The impact of technologies as well as the impact of variance are studied and presented. To quantify uncertainty, first, the sources of uncertainty in performance analysis must be established. Once these sources are identified, their effects are quantified by using the Probabilistic Certificate of Correctness (PCC) methodology; in which simulations are performed to establish confidence in predicted performance.

This report outlines the logic and assumptions made by the author to arrive at the current implementation of the NDARC Optimized Calibration (NDARC – OC) and NDARC Optimized Performance Spreadsheet (NDARC – OPS) tools. The objective of this work is to reduce the effort required by the user to calibrate a NDARC model against data from a comprehensive analysis tool, and not to remove subject matter experts from the process. Any feedback NDARC users is highly encouraged, and will be incorporated into future revisions of the tool.

4.2 Motivation

The NDARC Performance Spreadsheet was developed by Wayne Johnson to facilitate the process of calibrating NDARC's rotor power models against experimental or hider fidelity comprehensive analysis data. The spreadsheet consists of a set of NDARC coefficients and exponential factors that control the shape of NDARC's built-in polynomials, which estimate the induced power coefficient and profile drag coefficient of the rotor under the specified flight conditions. The calibration process aims to minimize the overall error between the NDARC models and higher fidelity comprehensive data for both of these coefficients, and is an essential task of using NDARC for rotorcraft design and performance analyses.

The rotor spreadsheet as distributed with NDARC requires the user to manually perform iterations by changing the NDARC variables, one at a time, until they are satisfied that the NDARC power models approximate the higher fidelity comprehensive data accurately enough. This leads to ambiguity in the results, as there is currently no direct way to quantify the accuracy of the results. Additionally, this process relies heavily on the user having an intimate knowledge of the behavior of the power model variables, and severely restricts exploration of the design space (made up of the different combinations of NDARC variables) as the manual iteration will almost certainly hone in on a single local minimum rather than finding the best global solution to minimizing the error. Finally, the use of manual iteration to perform this task is

incredibly inefficient, especially if the task is to be repeated many times for different sets of calibration data, requiring a lot of user effort and time.

To address these issues, the calibration of the NDARC variables can be formulated as an optimization problem. In doing so, the calibration process can be fully automated, requiring minimal user set up through a single input file. This optimization approach is described in the remainder of this report.

4.3   Problem Formulation

This section outlines the steps and assumptions that went into formulating the NDARC model calibration process into an optimization problem. First the calibration process is reviewed, identifying the NDARC variables that make up the design space and the approach used to quantify the error between the comprehensive calibration data and NDARC model. Justification is then provided for the selection of a genetic algorithm as the optimization algorithm, as well as a brief explanation for how the algorithm itself is implemented.

4.3.1   Calibration Process

Design Space

Within NDARC, the rotor power model is broken down into two independent design spaces: induced power and profile power. The induced power NDARC variables determine the calculation of the induced power coefficient (κ), while the profile power NDARC variables determine the calculation of the profile drag coefficient (C_D). The NDARC variables associated with the induced and profile power design spaces are provided in Table 4-1 and Table 4-2, respectively. Default values are also listed for each of the design variables, which will be addressed in Section 0.

The user should be aware that some of the NDARC variables listed in the two tables below represent physical values. The physical variables are the induced velocity factors (Ki_hover, Ki_climb, etc.), the variables with a CTs_ prefix, and the advance ratio variables (mu_ prefix). It is up to the user to set physically feasible values for these design variables. All other variables represent a coefficient or exponent of a polynomial curve fit and have no physical meaning.

As can be seen from these two tables, the design space of this problem has the potential to become quite large, encompassing over 30 design variables in each separate design space. In addition, almost all of the NDARC variables must be treated as continuous variables over some practical range of values, further increasing the complexity of the problem. To reduce the dimensionality, it is desired that the user have the ability to select which of the NDARC variables will be varied during the optimization process (henceforth referred to as "design variables"), and which NDARC variables will be held fixed during the optimization

algorithm (henceforth referred to as "constant parameters"). This drives the need for a flexible, scalable optimization algorithm. The selection of the optimization algorithm is discussed in detail in Section 4.3.2.

Table 4-1: Rotor induced power design variables

Description	Variable	Default Value
model (1 constant, 2 standard)	MODEL_ind	2
Induced velocity factors (ratio to momentum theory induced velocity)		
Hover	Ki_hover	1.12
Axial climb	Ki_climb	1.08
Axial cruise (propeller)	Ki_prop	2
Edgewise flight (helicopter)	Ki_edge	2
Variation with Thrust		
CT/s for Ki_h variation	CTs_Hind	0.08
Coefficient for Ki_h	kh1	0
Coefficient for Ki_h	kh2	0
Exponent for Ki_h	Xh2	2
CT/s for Ki_p variation	CTs_Pind	0.08
Coefficient for Ki_p	kp1	0
Coefficient for Ki_p	kp2	0
Exponent for Ki_p	Xp2	2
Variation with Shaft Angle		
Coefficient for Ki_p	kpa	0
Exponent for Ki_p	Xpa	2
Variation with Lift Offset		
Coefficient for f(offset)	ko1	0
Factor for f(offset)	ko2	8
Constant in Ki transition from hover to axial cruise	Maxial	1.176
Exponent in Ki transition from hover to axial cruise,	Xaxial	0.65
Variation with Axial Velocity		
Advance ratio for Ki_prop	mu_prop	1
Coefficient for Ki(muz) (linear)	ka1	0
Coefficient for Ki(muz) (quadratic)	ka2	0
Coefficient for Ki(muz)	ka3	0
Exponent for Ki(muz)	Xa	4.5
Variation with Edgewise Velocity		
Advance ratio for Ki_edge	mu_edge	0.35
Coefficient for Ki(mu) (linear)	ke1	0.8
Coefficient for Ki(mu) (quadratic)	ke2	0
Coefficient for Ki(mu)	ke3	1
Exponent for Ki(mu)	Xe	4.5
Variation with rotor drag	kea	0
Minimum Ki	Ki_min	1
Maximum Ki	Ki_max	10

Table 4-2: Rotor profile power design variables

Description	Variable	Default Value
Technology Factor		
Profile power	TECH_drag	1
Reference Reynolds number (0. for no correction)	Re_ref	0
Basic model (1 array, 2 equation)	MODEL_basic	2
Array (cd vs thrust-weighted blade loading)		
Number of points (maximum 25)	ncd	24
Equation		
CT/s for minimum profile drag	CTs_Dmin	0.07
Coefficient in drag vs CT/s function (constant for hover/edgewise)	d0_hel	0.009
Coefficient in drag vs CT/s function (constant for axial)	d0_prop	0.009
Coefficient in drag vs CT/s function (linear hover/edgewise)	d1_hel	0
Coefficient in drag vs CT/s function (linear for axial)	d1_prop	0
Coefficient in drag vs CT/s function (quadratic for hover/edgewise)	d2_hel	0.5
Coefficient in drag vs CT/s function (quadratic for axial)	d2_prop	0.5
Variation with shaft angle, coefficient for cdp	dprop	0
Variation with shaft angle, exponent for cdp	Xprop	2
CT/s for separation (Dcd = d(CT/s-CT/s_sep)^X)	CTs_sep	0.07
Factor in drag increment	dsep	4
Exponent in drag increment	Xsep	3
Variation with edgewise velocity, coefficient	df1	0
Variation with edgewise velocity, coefficient	df2	0
Variation with edgewise velocity, exponent	Xf	2
Stall model (0 none)	MODEL_stall	1
CT/s at stall (D=|CT/s|-f*CT/s_stall, Dcd=d1*D^X1+d2*D^X2)		
Number of points (maximum 20)	nstall	10
Constant in stall drag increment	fstall	1
Factor in stall drag increment	dstall1	2
Factor in stall drag increment	dstall2	40
Exponent in stall drag increment	Xstall1	2
Exponent in stall drag increment	Xstall2	3
Variation with Lift Offset		
Coefficient for f(offset)	do1	0
Factor for f(offset)	do2	8
Variation with rotor drag	dsa	0
Compressibility model (0 none, 1 drag divergence, 2 similarity)	MODEL_comp	1
Similarity Model		
Factor	fSim	1
Blade tip thickness-to-chord ratio	thick_tip	0.08
Drag Divergence Model (D=(Mat-Mdd), Dcd=d1*D+d2*D^X)		
Coefficient in drag increment	dm1	0.056
Coefficient in drag increment	dm2	0.416
Exponent in drag increment	Xm	2
Drag Divergence Mach Number (Mdd = Mdd0 - Mddcl*cl)		

Mdd at zero lift	Mdd0	0.88
Derivative with lift	Mddcl	0.16

Required Calibration Data

The information required from a comprehensive analysis tool to calibrate a NDARC model is provided in Table 4-3. The information includes four independent variables (mux, muz, CT/, offset, and MAT), and two dependent variables (values for profile drag and induced power coefficients from the comprehensive analysis tool). A "calibration data point" refers to a single, unique set of the independent variables and associated dependent variable values, while the "calibration data set" refers to the collection of all calibration data points. A single NDARC model is calibrated against the entire calibration data set, which may consist of any number of calibration data points.

Table 4-3: Information required to calibrate NDARC models

Variable	Description
mux	Advance ratio along the x-axis
muz	Advance ratio along the z-axis
CT/s	Blade Loading (thrust coefficient / solidity)
MAT	Maximum Mach number at the advancing tip
Offset	Design lift offset value
Cd	Profile drag coefficient
Kappa	Induced power coefficient

Calculation of Calibration Error

Error calculations for both the induced power coefficient and profile drag coefficient serve as the objective functions to be minimized during the optimization. For both coefficients, the total error is calculated as the sum of the absolute relative error (summed over *N* calibration data points), where the value estimated from the NDARC curve fits is measured relative to the true value provided by either a comprehensive analysis tool (such as RCAS or CAMRAD) or some other form of higher fidelity data. As the number of calibration data points, *N*, may vary from case to case, the objective functions in the optimization problem are represented as the average of this total error calculation, as shown in the equation below. This approach provides a metric to measure the calibration accuracy that is independent of the number of calibration data points used (i.e. the magnitude of the error does not scale directly with the number of calibration data points).

$$C_D \, Obj. \, Function = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \left| \frac{C_{D_{est}} - C_{D_{true}}}{C_{D_{true}}} \right| \qquad \kappa \, Obj. \, Function = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \left| \frac{\kappa_{est} - \kappa_{true}}{\kappa_{true}} \right|$$

4.3.2 Selection of Optimization Algorithm

The nature of the design space drove several requirements for the selection of the optimization algorithm:

- Algorithm must maintain efficiency when scaling to handle large design space
- Potentially handle both discrete and continuous design variables
- Must be capable of efficiently exploring a multi-modal design space

Gradient based optimization algorithms were ruled out due to their inability to efficiently explore design spaces that are highly multi-modal. This led to the investigation of metaheuristic algorithms, which tend to exhibit better global optimization properties for multi-modal design spaces. The need for the optimization algorithm to efficiently scale to large design spaces while potentially handling both continuous and discrete variables led the selection of a genetic algorithm. The genetic algorithm implemented for the NDARC calibration process is described in the following subsection.

Genetic Algorithm for NDARC Calibration

A semi-elitist genetic algorithm was implemented for the NDARC calibration process. The genetic algorithm first requires that all design variables be discretized into a base-2 binary number. The number of bits required to represent a given design variable can be calculated using Equation 1, which is dependent on both the desired range and resolution of a given design variable. An "individual" in the genetic algorithm describes a unique design (i.e. combination of design variables within the feasible design space). Each individual is represented in the base-2 system by a single "chromosome", which is simply the concatenation of the binary strings of all design variables. A notional example of an individual's chromosome is illustrated in Figure 4-1 for a system with three design variables.

$$N_{bits} = \frac{\ln\big((Range/Resolution)+1\big)}{\ln(2)}$$

Equation 1
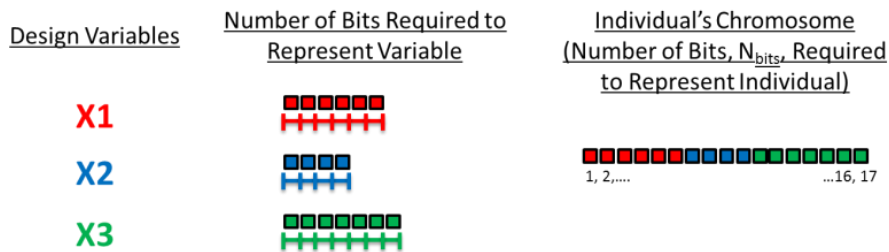
Figure 4-1: Composition of an individual of the genetic algorithm

The implementation of the genetic algorithm is outlined in Figure 4-2. It begins with the generation of an initial "parent population", which is a collection of *N* individuals within the feasible design space. The next

step requires that individuals from the parent population be selected for reproduction. Selection is carried out through a deterministic tournament, where two individuals are randomly sampled with replacement from the parent population. As the aim of the NDARC calibration is to minimize the error between the NDARC model and calibration data, the parent with the lower calibration error has the higher rank, and is thus selected as the preferred parent for reproduction. The tournament selection is performed N times, such that $N/2$ pairs of parents have been selected. Next, reproduction occurs in which each pair of parents generates two new individuals, or "children", which occurs through a combination of crossover and mutation of the parent chromosomes. The result of the reproduction step is a child population of N individuals. The next generation begins by establishing the new parent population, which consists of the top 10% of the parent individuals (to enforce elitism) and top 90% of the child individuals (to encourage exploration).
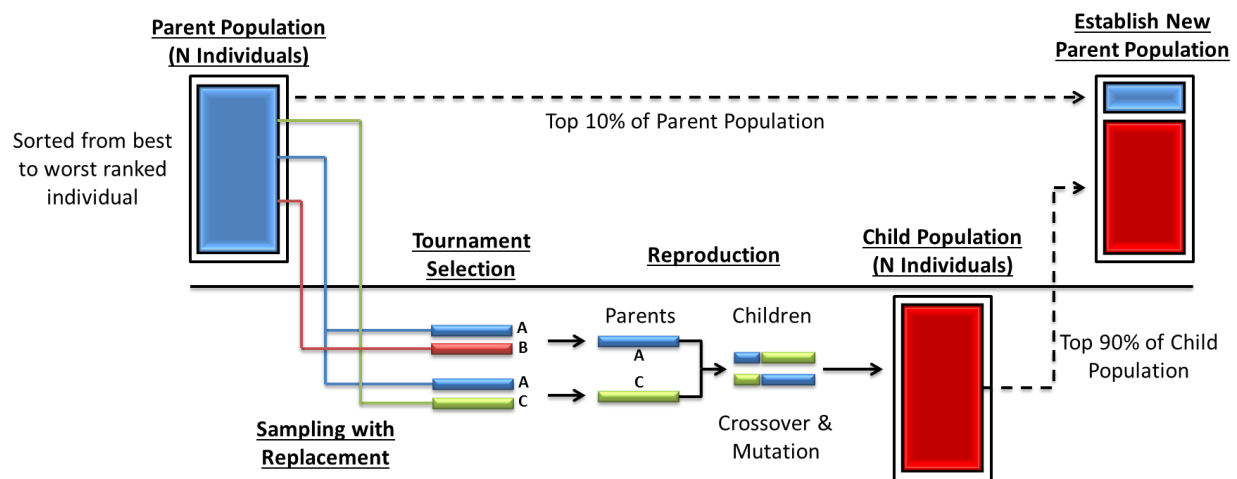
Figure 4-2: Single iteration of genetic algorithm for NDARC calibration process

Handling Discrete Variables

As genetic algorithms necessitate that all continuous design variables be discretized, the algorithm can easily handle both the discrete variables (such as "MODEL_ind" from Table 4-1) and continuous design variables, while also allowing the user to easily set bounds on the allowable range for each variable. The latter characteristic requires the user to have some knowledge of the NDARC design space, such that realistic ranges are set for each variable, a requirement that simply cannot be avoided.

Scaling the Genetic Algorithm

The calibration of a NDARC model suffers from the "curse of dimensionality", in that the size of the design space grows exponentially as the number of, and allowable range of the design variables increases. Because of this, there is not one global setting for the genetic algorithm (in terms of number of individuals in a population and number of generations to iterate through) that allows for an efficient and effective exploration of all possible design spaces. To address this issue, the NDARC genetic algorithm is scaled on a case by case basis by using linear scaling factors to vary the number of individuals (N_{ind}) in each population and the number of generations (N_{gen}) that the genetic algorithm is run to based on the estimated size of the design space. The number of bits in an individual's chromosome (N_{bits}) serves as representation for the size of the design space. The number of individuals that make up a population is then calculated as the product of N_{bits} and the population scaling factor (*popFactor*). The number of generations that the algorithm is run to is determined as the product of N_{ind} and the generation scaling factor (*genFactor*). This process is demonstrated in Figure 4-3.
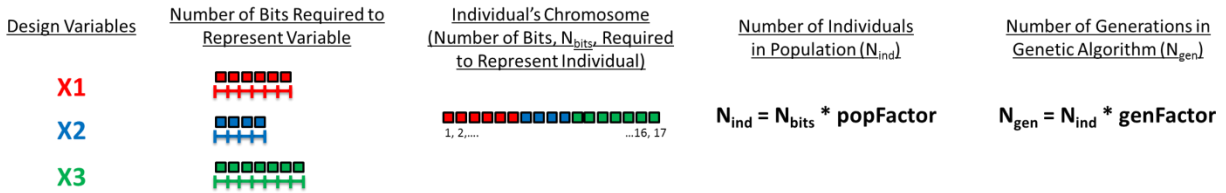
Figure 4-3: Scaling the number of individuals and generations of the genetic algorithm based on the size of the design space

Despite the effectiveness of this approach, the "curse of dimensionality" begins to dominate the optimization problem as the design space grows in size. That is, as the number of design variables increases, the size of the design space grows exponentially while the explorative parameters of the genetic algorithm (N_{ind} and N_{gen}) only scale linearly. At first thought, one might suggest scaling the exploration parameters exponentially as well. However, doing so makes the run times prohibitively expensive, and is thus not a viable solution. As an alternative to this approach, the optimization algorithm has been written such that the linear scaling factors are inputs to the genetic algorithm (as discussed in Section 2593088.0.-946779886). Thus, for larger design spaces the magnitude of the scaling factors can be increased to account for the increased dimensionality of the problem, at the cost of computational expense. At this point the calibration of the NDARC models becomes a tradeoff that the user must make between design space exploration and execution time. Despite this drawback, the case study discussed in Section 4.6 demonstrated that the NDARC – OC tool is capable of arriving at a better solution in a more efficient manner then could be obtained through manual iterations.

4.3.3 Selection of Programming Language

Two key criteria went into the selection of the programming language for this process: 1) the code must be open source, and 2) it must be capable of being compiled into an executable file that can be run on any machine without the need for an interpreter. The requirement that the code be open source ruled out the use of common engineering tools with optimization capabilities such as Model Center and MATLAB. Because of this, consideration was given to developing the optimization process in both Python v3.4 and Fortran 95. Python was ultimately selected due to its ease of integration with the NDARC – OPS and its ability to meet the two criteria stated above. However, the design space of this problem has the potential to grow to the point that it is computationally prohibitive to run a full analysis in Python. If this becomes a major roadblock in the use of the NDARC calibration tools, the optimization code can be transitioned to Fortran for computational efficiency, requiring minimal restructuring of the optimization process.

4.4 Implementation of NDARC – OC Tool

The use of optimization techniques to automate the calibration of NDARC models relies on two things: the set of available NDARC design variables is fixed, and the calibration data set is known and can be provided in some structured format. With this information, enough structure is provided to allow the entire process to be automated, requiring minimal user set up while providing fast, accurate results given that the information provided is appropriate. An overview of the new calibration process is provided in Figure 4-4, which requires the three general steps described below.

1. In the input files, set: which NDARC variables are design variables for the optimization process versus constant parameters, the calibration data, and the run settings
2. Run the optimization algorithm
3. Analyze results, and if necessary make adjustments to design space and re-run the optimization
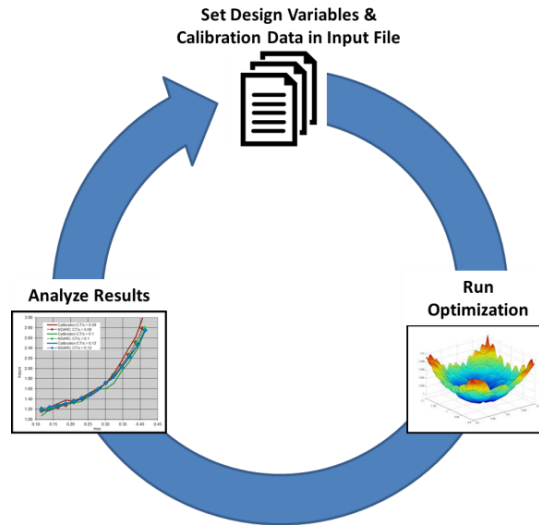
Figure 4-4: Automation of calibration process using optimization technique

4.4.1 NDARC Calibration Input File

A single input file is required to run the NDARC – OC tool. This input file MUST be called "NDARC Calibration Settings.inp", and it must be located in the "Input Files" folder shown in Figure 4-8 in order for the NDARC optimization code to find and access the file. The calibration input file is organized similarly to a Fortran style NAMELISTs input file (but it is not a true NAMELIST input file). All NAMELIST group names must be preceded by an ampersand ("&"), and must be in all caps. The NAMELISTs may occur in any order. The available NAMELISTs are described in Table 4-4, and the inputs available to each NAMELIST are described in the following sections. For an example input file, please refer to Appendix C.

Table 4-4: Available NAMELISTs in NDARC Calibration Settings input file

NAMELIST	Description
&RUN_SETTINGS	Specify desired run settings
&INDUCED_NDARC_VARIABLES	Set default values for constant parameters, or range/resolution
&PROFILE_NDARC_VARIABLES	for design variables
&CALIBRATION_DATA_SET	Set data from comprehensive analysis tool to calibrate NDARC model against

Several requirements must also be followed when setting input variables in the input file.

- All input variable names are case sensitive and must be spelled correctly
- Variable names should be separated from the input data by an equals sign (e.g. Ki_hover = 1)
- Any input variable that contains more than one value should separate these values using a comma (e.g. Ki_hover = 1, 3, .1)

RUN_SETTINGS NAMELIST

Currently, six settings are available in the &RUN_SETTINGS NAMELIST as described in Table 4-5, along with default values if no input is provided. If *NumRuns* is set to a value greater than 1, then the optimization algorithm will optimize an NDARC model using the current settings *NumRuns* times, and will record the NDARC model with the lowest error for both the induced and profile power design spaces. The intent of this setting is to allow the user to run the optimization several times to reduce the chance that the genetic algorithm gets stuck in a local minima, without requiring the user to be present to manually re-run the optimization code themselves.

Table 4-5: Input variables available in the &RUN_SETTINGS NAMELIST

Variable Name	Description	Default Value
saveAsFilename	Specify filename to save output files to	NDARC Optimized Model
numRuns	Set the number of runs performed during the optimization (should be integer value)	1
inducedPopulationFactor	Factor that scales the size of the population in the genetic algorithm uses to optimize the induced power design space	4
inducedGenerationFactor	Factor that scales the number of generations the genetic algorithm uses to optimize the induced power design space	5
profilePopulationFactor	Factor that scales the size of the population in the genetic algorithm uses to optimize the profile power design space	4
profileGenerationFactor	Factor that scales the number of generations the genetic algorithm uses to optimize the profile power design space	5

INDUCED/PROFILE_NDARC_VARIABLES NAMELISTs

The variables available for the induced power and profile power design spaces are the same as those listed in Table 4-1 and Table 4-2, respectively. The input variables can be listed in any order. However, all variables must be spelled correctly with the correct capitalization. Table 4-6 shows the proper format for setting a constant parameter, design variable, and a match variable. Again, the variable name should be separated from the input values by an equals sign, and for design variables the input values should all be separated by a comma. The match variable type requires that the current variable always have the same value as another variable during the optimization. The example listed in Table 4-6 requires that "Ki_hover" always have the same value as "Ki_edge", even if "Ki_edge" is varied during the optimization runs. In addition, if a variable is not found in the proper NAMELIST (or is simply omitted from the input file), then it will be set to the default values listed in Table 4-1 or Table 4-2.

Table 4-6: Formatting design variable versus constant parameters in the induced/profile NAMELISTS

Variable Type	Expected Input Order	Example
Constant Parameter	Constant Value	Ki_hover = 1.1
Design Variable	Lower Bound, Upper Bound, Resolution	Ki_hover = 1, 3, .1
Match Variable	String containing variable name that current variable must have same value as	Ki_hover = Ki_edge

CALIBRATION_DATA_SET NAMELIST

The $CALIBRATION_DATA_SET NAMELIST has no variable declarations, but instead expects a list of information. The list should be in table format, with columns corresponding to the independent/dependent variables listed in Table 4-3, and the rows corresponding to individual calibration data points. In each row, data must be entered in the following order: mux, muz, CT/s, MAT, κ, and C_D, and all data entries must be separated by a comma. If the user does not wish to calibrate against one of the dependent variables under certain conditions, then a value of "0" should be entered for that dependent variable. An example of this is shown in Figure 4-5, where the last three calibration data points are only being used to calibrate the induced power coefficient, kappa.

```
&CALIBRATION_DATA_SET
#  mu muz CT/s    MAT  Offset Actual_Kappa Actual_Cd
  .1687, 0, .06998, .7203, 0, 1.0706, .00857
  .1927, 0, .06997, .7351, 0, 1.1432, .00859
  .2167, 0, .07002, .7499, 0, 1.1412, .00859
  .2407, 0, .06994, .7647, 0, 1.3486, .00865
  .2645, 0, .07, .7794, 0, 1.4106, .00869
  .2883, 0, .0699, .7941, 0, 1.6112, .00869
  .312, 0, .06997, .8088, 0, 1.8293, .00884
  .3354, 0, .07008, .8233, 0, 2.0511, .0091
  .3587, 0, .06997, .8378, 0, 2.3868, .00936
```

Figure 4-5: Example showing format of &CALIBRATION_DATA_SET NAMELIST

The implementation of the calibration data in this manner adds flexibility to the spreadsheet, as the user can now quickly change the calibration data and run the optimization with very little effort. However, the code is limited to calibration data in this specific format. If for any reason the type of calibration data must be changed (i.e. no longer calculating values for drag, but some other parameter), the NDARC – OC will have to be altered to reflect this.
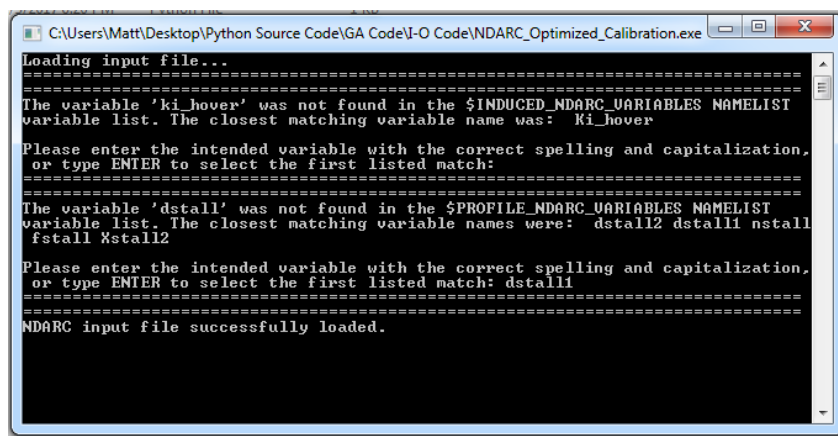
4.4.2   Input File Error Checking

Error checks have been built into the optimization code to ensure that the input file has been formatted properly before running the optimization. Two main checks are performed: 1) The variable names are checked against a list of possible input file variables to prevent spelling or capitalization errors, and   2) The values associated with each variable are checked to make sure that no logical rules are broken and that the correct number of input values is provided. Checking the variable name spelling is important because the optimization code heavily utilizes the Python dictionary variable type, which accesses fields within the

76

dictionary using the NDARC variable name strings. The input value checking ensures that the correct information has been provided and in the proper format to properly set up the optimization problem. Both error checks are elaborated on in the following sections.

Checking Variable Names

Every variable name read from the input file is automatically checked against the list of available input variables. If the variable name does not match a known variable of the current NAMELIST exactly (spelling and capitalization), then the NDARC – OC tool will automatically alert the user and suggest the closest found matches within the current NAMELIST as alternatives. The user then has the ability to re-type the intended variable into the command prompt or select the closest suggested match, without having to end the optimization run and fix the input file directly. For this to work, the intended input variable MUST be in the proper NAMELIST, as other NAMELISTs are not checked if the variable was not found. Once all variable name errors are corrected, the code will indicate that the NDARC input file was successfully loaded, and the optimization will begin. A demonstration of this is shown in Figure 4-6.

Figure 4-6: NDARC - OC tool automatically checking variable names from the input file to make sure that they match a known variable exactly

Error Report

In addition to the two output files listed Section 4.4.4, a file called "ERROR REPORT.out" is written to the "Output Files" folder after each run. This file contains information regarding the incorrect formatting of the input data for all NAMELISTs. Unlike with the variable names, no "best guess" can be made for the intended value for each input variable. Thus, if an error for the NDARC variable values is found (i.e. incorrect number of inputs listed for a given variable), then the code is aborted before the optimization begins and an error message is written to the output file. There are three input file format errors that have been accounted for; these three errors are described in Table 4-7. In addition to the error message, the error

report will contain information regarding what NDARC variable caused this error (this does not apply for errors in the calibration data set table), as well as the line of the input file where the specified error occurred. If a known error occurs, the code will finish reading the input file to determine all formatting errors before printing the final "ERROR REPORT.out" file and aborting. An example of this error report is shown in Figure 4-7. The input file associated with this error report is provided in Appendix F with the input errors highlighted in yellow.

If the NDARC – OC code crashes unexpectedly, the file will return "NDARC-OC crashed unexpectedly", and the code will abort immediately. If this occurs, please save the input file and contact Eric Spero so that the issue can be addressed. If the run is completed successfully, the "ERROR REPORT.out" file will simply print "RUN COMPLETED SUCCESSFULLY".

Table 4-7: Error messages in Error Report that occur when reading input file

Error Message	Description
DESIGN VARIABLE BOUND ERROR	The lower bound entered for a design variable is greater than or equal to the upper bound.
NDARC VARIABLE INPUT ERROR	The user has listed the incorrect number of inputs for an NDARC variable. One input should be listed for a constant parameter, and three should be listed for a design variable.
CALIBRATION DATA INPUT ERROR	The user has listed an incorrect number of inputs for the calibration data set table. Six values should be entered in each row, as addressed in Figure 4-5.
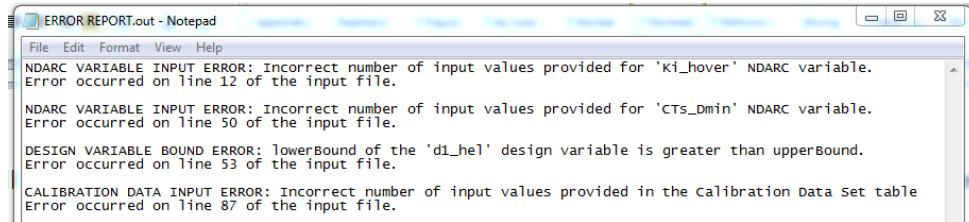
Figure 4-7: Example of error report from input file with multiple input formatting issues (input file in Appendix F)

4.4.3 Running Optimization Algorithm

The optimization code is written in a single Python file called "NDARC_Optimized_Calibration.py", which contains all the functions required to run the optimization process. This python file, along with the "Input Files" folder, and the "Output Files" folder must all be located in the same folder or directory location, as shown in Figure 4-8. To run the python file directly, the following command should be entered into the command prompt after navigating to the directory that contains the "NDARC_Optimized_Calibration.py"

file (assuming that "python" is a windows environment variable that refers to the python interpreter): *python NDARC_Optimized_Calibration*

Figure 4-8: Required contents of the "NDARC Optimized Calibration" folder

Runtime User Displays

Once the NDARC – OC code is run, a command window will appear providing the user with information regarding the progress of the optimization algorithm. The initial display indicates to the user: the design space that is currently being optimized (induced or profile power design space), the run number that is currently being executed, the number of generations that the genetic algorithm will be run to, and the number of individuals in each population. An example of this is shown below in Figure 4-9. This display occurs every time that a new run has been started.
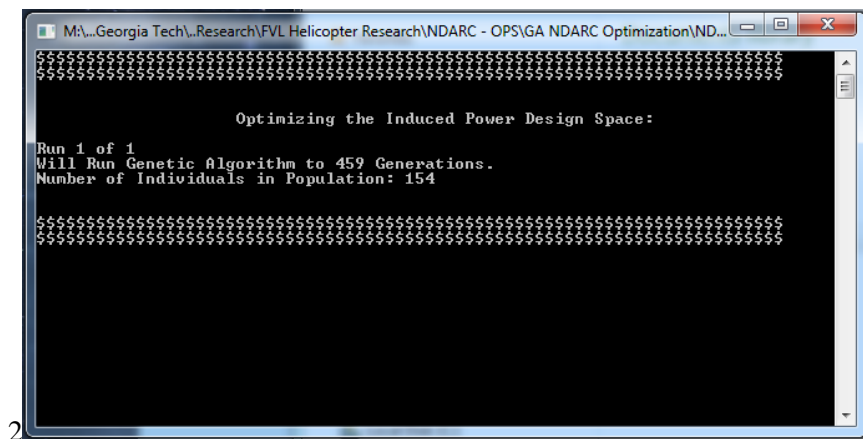
Figure 4-9: Initial display to user when optimization executable is launched

During execution of the optimization algorithm, the display will indicate the current generation, the relative percent error of the current best configuration, and the estimated time remaining. It should be noted that this estimated time remaining is ONLY in reference to the current run.

Figure 4-10: User display during the execution of the optimization algorithm

The user also has the option to abort a run early by typing "Ctrl+C" into the command prompt once. This will abort the current run, but will still save the top ranked configuration from this run. For instance, if the optimization algorithm is currently on Run 1 of 2 in the profile drag coefficient design space, then typing "CTRL+C" will end Run 1, and Run 2 of the profile drag coefficient design space will begin immediately after. To abort the entire optimization algorithm, type "Ctrl+C" twice within a three second span, or simply close out of the window directly. This will prevent any results from being written to the "Output Files" folder.
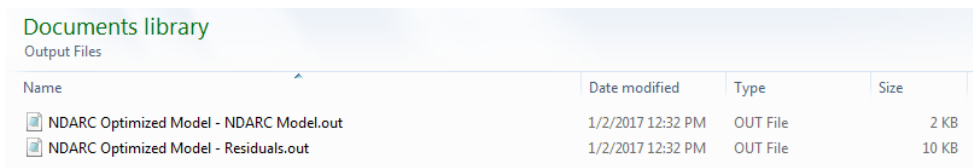
Figure 4-11: User display upon aborting the current run

4.4.4   NDARC Calibration Output Files

Two separate output files are written by the optimization code; one contains the NDARC model with all NDARC variables and associated values, and the other contains a table with the calibration data, calculated values for the induced power and profile drag coefficients, and the errors associated with each calibration data point. Both files are written to the "Output Files" folder in the location as shown in Figure 4-8. The "SaveAsFilename" variable from the &RUN_SETTINGS NAMELIST is used as the base name for each

file, with an additional string attached to differentiate between the residuals file ("Residuals.out") and the NDARC model file ("NDARC Model.out"). Sample file names are provided in Figure 4-12, while sample output files can be found in Appendix D & E

Figure 4-12: Sample output file name formats

4.4.5 Limitations of NDARC – OC Tool

As discussed in Section 2593088.0.-946779886 the calibration of NDARC models using this approach suffers from the "curse of dimensionality". This issue is encountered whether the calibration is done through the NDARC – OC tool, or by manual iterations. Though the genetic algorithm used within the NDARC – OC is scaled in an attempt to account for the increase in dimensionality that occurs as the number of NDARC variables considered increases, due to computational limitations, it is not possible to scale the algorithm exponentially with the design space. Thus, if the size of the design space is extremely large (considering over ≈15-20 design variable simultaneously) the performance of the NDARC – OC tool will likely be degraded. It has been found that this is especially true for the profile drag coefficient design space.

Because of this it is highly suggested that users does not to attempt to optimize the entire design space (for both induced and profile power) in a single optimization run. Rather, judgement should be made in selecting a representative set of NDARC variables as design variables. An alternative solution is to perform the optimization in multiple steps, optimizing a subset of the design space during each optimization run. Though this is not ideal, this approach can produce acceptable results in a reasonable amount of runtime.

An alternative to the approach stated above is to set up the NDARC – OC code to run overnight, while using large values for the population factors to account for extremely large design spaces. Successful calibrations have been performed using all NDARC variables as design variables, while using population factors on the order of 40~50 with a generation factor of 1. These runs are obviously extremely expensive to perform, often taking over 6 hours for a single run, but they have been found to provide good results. After completing such a run, the calibration can be fine-tuned by using the NDARC variable values found from the long run as the constant parameter values, and then performing smaller optimization runs on a subset of the original NDARC design variables.

4.5 Implementation of NDARC – OPS Tool

The NDARC – Optimized Performance Spreadsheet tool is an extension of the NDARC – Optimized Calibration (NDARC – OC) tool for Windows OS users. The NDARC – OPS took makes use of Microsoft Excel as a GUI for running the NDARC – OC tool. The NDARC – OPS process (shown in Figure 4-13) is nearly identical to the process described in Figure 4-4 for the NDARC-OC tool. The only difference is that the design variables and calibration data is now set in the Excel sheet, and the process of writing the NDARC – OC input files is automated using VBA. This, in general, should make it easier to run the calibration process for Windows users, and reduce user errors in formatting the NDARC – OC input file. In addition to automatically writing the input files, the NDARC – OPS also provides several plots to help visualize the accuracy of the current model, as will be discussed in Section 4.5.3.
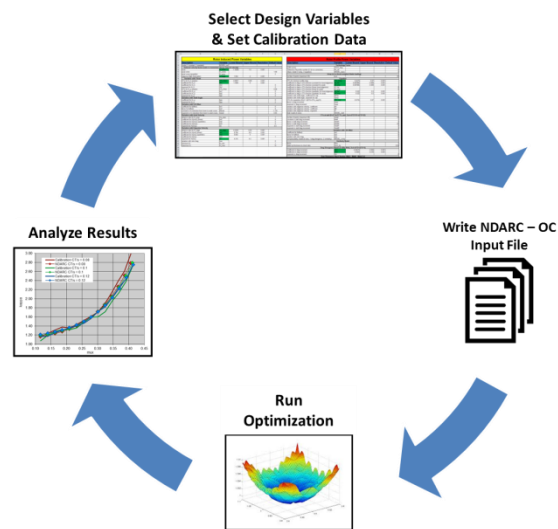
Figure 4-13: NDARC - OPS optimization process

To allow for the process described above to be automated, the NDARC – OPS must be located in a folder with the "NDARC Optimized Calibration" folder. The "NDARC Optimized Calibration" folder must have the same contents as described in Figure 4-8. This allows the NDARC – OPS to write the input files to the correct location, as well as extract the output files after the calibration process has finished. The remainder of this section described the proper use of the NDARC – OPS user interface.
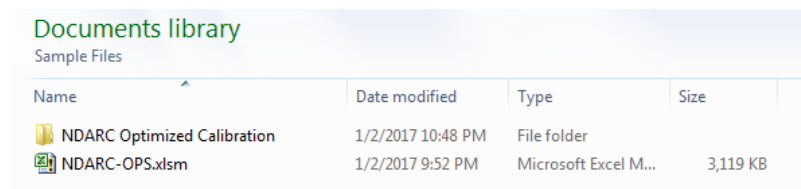
Figure 4-14: Proper location of "NDARC Optimization" folder relative to NDARC – OPS

4.5.1 Setting Calibration Data

The calibration data (from comprehensive analysis tools such as CAMRAD or RCAS) is set on the "Calibration Data Sets" sheet of the NDARC – OPS. The information required in this table is the same that was outlined in Table 4-3 and Figure 4-5. The only difference between the table shown in Figure 4-15 and the previous examples is the first column labeled "Case". This column is purely for the user to be able to group sets of calibration data points by different case numbers for post-processing of results within the NDARC - OPS, and will have no effect on the optimization itself (this column is not written to the NDARC – OC input file). The VBA code within the NDARC – OPS will pull the data out of this specific table. The table can be of arbitrary length (the code will read the calibration tables until it has found a blank row with no data in it), but the column order MUST be followed exactly.

Calibration Data Set

Case	mu	muz	CT/s	MAT	Offset	Actual_Kappa	Actual_Cd
1	0	0	0.05997	0.6163	0	1.0158	0.00837
1	0	0	0.06894	0.6163	0	1.0549	0.00842
1	0	0	0.07814	0.6163	0	1.0887	0.0085
1	0	0	0.08744	0.6163	0	1.1206	0.00867
1	0	0	0.0969	0.6163	0	1.1486	0.00888
1	0	0	0.10656	0.6163	0	1.1724	0.00916
1	0	0	0.11641	0.6163	0	1.1922	0.00963
1	0	0	0.12644	0.6163	0	1.2064	0.01075
1	0	0	0.13666	0.6163	0	1.2172	0.01196
1	0	0	0.14685	0.6163	0	1.228	0.0134
1	0	0	0.15688	0.6163	0	1.2364	0.0152
1	0	0	0.16665	0.6163	0	1.2408	0.01786
1	0	0	0.1761	0.6163	0	1.2431	0.02135
1	0	0	0.1761	0.6163	0	1.2431	0.02135
1	0	0	0.19258	0.6163	0	1.2353	0.035
1	0	0	0.1992	0.6163	0	1.2323	0.04284
1	0	0	0.20479	0.6163	0	1.2289	0.05138
1	0	0	0.20981	0.6163	0	1.2225	0.06283
1	0	0	0.21415	0.6163	0	1.2181	0.07734
2	0.1687	0	0.06998	0.7203	0	1.0706	0.00857
2	0.1927	0	0.06997	0.7351	0	1.1432	0.00859
2	0.2167	0	0.07002	0.7499	0	1.1412	0.00859
2	0.2407	0	0.06994	0.7647	0	1.3486	0.00865
2	0.2645	0	0.07	0.7794	0	1.4106	0.00869
2	0.2883	0	0.0699	0.7941	0	1.6112	0.00869
2	0.312	0	0.06997	0.8088	0	1.8293	0.00884
2	0.3354	0	0.07008	0.8233	0	2.0511	0.0091

Variable Key

Variable	Description
mux	Advance ratio along the x-axis
muz	Advance ratio along the z-axis
CT/s	Blade loading (thrust coefficient / solidity)
MAT	Maximum Mach number at the advancing tip
Offset	Design lift offset value
Actual Cd	Actual profile drag coefficient from comprehensive analysis data
Actual Kappa	Actual induced power coefficient from comprehensive analysis data

Figure 4-15: Calibration data set table used to structure the information for the NDARC – OC tool

4.5.2 Setting Design Variables

The user interaction required to set up and run the optimization is contained within the "Optimization Set Up" sheet of the NDARC – OPS, which is labeled below in Figure 4-16. This interface provides the user with the following capabilities:

- Set the value of all $RUN_SETTINGS NAMELIST variables
- Set the values of each NDARC variable
- Ability to change which NDARC variables will be design variables (to be varied during the optimization) versus constant parameters

- A "Run Optimization" button that calls the NDARC – OC tool to run the entire optimization process based on the information in the current spreadsheet.

Figure 4-16: "Optimization Set Up" sheet of NDARC – OPS used to set up optimization problem

As noted in Figure 4-16, the current design variables of the optimization problem have a green shaded background in the spreadsheet, while all constant parameters have white backgrounds. To change a variable between a design variable and a constant parameter, the user simply has to double click on the variable name itself, as clearly specified in Figure 4-17.

In addition, the values that the user must set for each variable are dependent on the type of variable. Constant parameters require only a fixed value to be set, which is simply the constant value they will be held at during the optimization process. For design variables, the genetic algorithm requires that three values be provided: a lower bound, upper bound, and a resolution. To make it clear to the user what values should be provided, only the necessary inputs for each variable are visible. This is clearly shown in Figure 4-16, where the design variables have values visible in the "Lower Bound", "Upper Bound", and "Resolution" columns, while the constant parameters only have values visible in the "Fixed Value" column.

Rotor Induced Power Variables

Description	Variable	Lower Bound	Upper Bound	Resolution	Fixed Value
model (1 constant, 2 standard)	MODEL_ind				2
Induced velocity factors (ratio to momentum theory induced velocity)					
hover	Ki_hover	1	1.3	0.001	
axial climb	Ki_climb				1.08
axial cruise (propeller)	Ki_prop				Ki_edge
edgewise flight (helicopter)	Ki_edge	1	3	0.001	
Variation with Thrust					
CT/s for Ki_h variation	CTs_Hind	0	0.1	0.001	
coefficient for Ki_h	kh1	-8	8	0.001	
coefficient for Ki_h	kh2	-12	12	0.001	
exponent for Ki_h	Xh2				1.28
CT/s for Ki_p variation	CTs_Pind	0	0.1	0.001	
coefficient for Ki_p	kp1				kh1
coefficient for Ki_p	kp2				kh2
exponent for Ki_p	Xh2				Xh2
Variation with Shaft Angle					
coefficient for Ki_p	kpa				0
exponent for Ki_p	Xpa				3.84
Variation with Lift Offset					
coefficient for f(offset)	ko1				
factor for f(offset)	ko2				
constant in Ki transition from hover to axial cruise	Maxial				
exponent in Ki transition from hover to axial cruise	Xaxial				
Variation with Axial Velocity					
advance ratio for Ki_prop	mu_prop				
coefficient for Ki(muz) (linear)	ka1				
coefficient for Ki(muz) (quadratic)	ka2				
coefficient for Ki(muz)	ka3				0
exponent for Ki(muz)	Xa				4.5
Variation with Edgewise Velocity					
advance ratio for Ki_edge	mu_edge	0	0.45	0.001	
coefficient for Ki(mu) (linear)	ke1	0	5	0.001	
coefficient for Ki(mu) (quadratic)	ke2	0	5	0.001	
coefficient for Ki(mu)	ke3				3.798
exponent for Ki(mu)	Xe	0	8	0.001	
variation with rotor drag	kea				0
minimum Ki	Ki_min				1
maximum Ki	Ki_max				10

Rotor Profile Power Variables

Description	Variable	Lower Bound	Upper Bound	Resolution	Fixed Value
Technology Factor					
profile power	TECH_drag				1
Reference Reynolds number (0. for no correction)	Re_ref				0
Basic model (1 array, 2 equation)	MODEL_basic				2
Array (cd vs thrust-weighted blade loading)					
number of points (maximum 25)	ncd				24
Equation					
CT/s for minimum profile drag	CTs_Dmin	0	0.1	0.001	
coefficient in drag vs CT/s function (constant for hover/edge...	d0_hel	0	2	0.001	
coefficient in drag vs CT/s function (constant for axial)	d0_prop	0	0.1	0.001	
coefficient in drag vs CT/s function (linear hover/edgewise)	d1_hel				0
coefficient in drag vs CT/s function (linear for axial)	d1_prop				d1_hel
coefficient in drag vs CT/s function (quadratic for hover/edge)	d2_hel	0	2	0.001	
coefficient in drag vs CT/s function (quadratic for axial)	d2_prop	0	1	0.001	
variation with shaft angle, coefficient for cdp	dprop				0
variation with shaft angle, exponent for cdp	Xprop				0
CT/s for separation (Dcd = d(CT/s-CTs_sep)^X)	CTs_sep	0.035	0.15	0.001	
factor in drag increment	dsep				1.392
exponent in drag increment	Xsep				2.771
...city, coefficient	df1				0.002
...city, coefficient	df2				3.471
...city, exponent	Xf				9.884
	MODEL_stall				1
...stall, Dcd=d1*X1+d2*X2)	nstall				10
	fstall				1.274
factor in stall drag increment	dstall1				0.015
factor in stall drag increment	dstall2				107
exponent in stall drag increment	Xstall1				0.62
exponent in stall drag increment	Xstall2				2.707
Variation with Lift Offset					
coefficient for f(offset)	do1				0
factor for f(offset)	do2				0
variation with rotor drag	dsa				0
Compressibility model (0 none, 1 drag divergence, 2 similarity)	MODEL_comp				1
Similarity Model					
factor	fSim				1
blade tip thickness-to-chord ratio	thick_tip				0.08
Drag Divergence Model (D=(Mat-Mdd), Dcd=d1*D+d2*D*X)					
coefficient in drag increment	dm1	0	2	0.001	
coefficient in drag increment	dm2	0	4	0.001	
exponent in drag increment	Xm				3.694
Drag Divergence Mach Number (Mdd = Mdd0 - Mddcl*cl)					
Mdd at zero lift	Mdd0	0	2	0.001	
derivative with lift	Mddcl	0	2	0.001	

Callout: Double click to change between design variable and constant parameter.

Figure 4-17: Demonstrating how to change variable type between design variable and constant parameter

A limitation of this process is that a value must be provided for every column of a design variable, as the VBA code is reading these values and has no logic embedded within it to assign values to variables if they are missing from the spreadsheet table. That is, if a variable is a design variable, then the user must input a value for the "Lower Bound", "Upper Bound", and "Resolution". The "Fixed Value" is hidden from the user for the design variables, as it is not required for the optimization algorithm, but the current "Fixed Value" does not need to be deleted; it can be left as is and its value will just be hidden from view. Because of this, checks have been built into the VBA script to ensure that the proper values have been assigned. Upon clicking the "Run Optimization" button, the VBA code will check all of the inputs, and provide alert messages if any input values are missing. A few examples of this are shown below in Figure 4-18. The alerts will tell the user what variable to look at, what table the variable is in (either induced or profile power), and it will select the cell that needs to be changed.

Likewise, for a constant parameter the values for the "Lower Bound", "Upper Bound", and "Resolution" will be hidden from the user (but their values will not be deleted). However, a constant parameter does not require a value to be set. If the "Fixed Value" column is left blank for a constant parameter, then the default value for that NDARC variable (based on values from Table 4-1 and Table 4-2) will be assigned to it for the entire optimization run. Additionally, if a string representing a different variable is input in the "Fixed Value" column, then that NDARC variable will be treated as a match variable during the optimization run. For example, in Figure 4-17 shown above, the value for "kp1" will always be the same as the value assigned for "kh1" during the optimization runs.
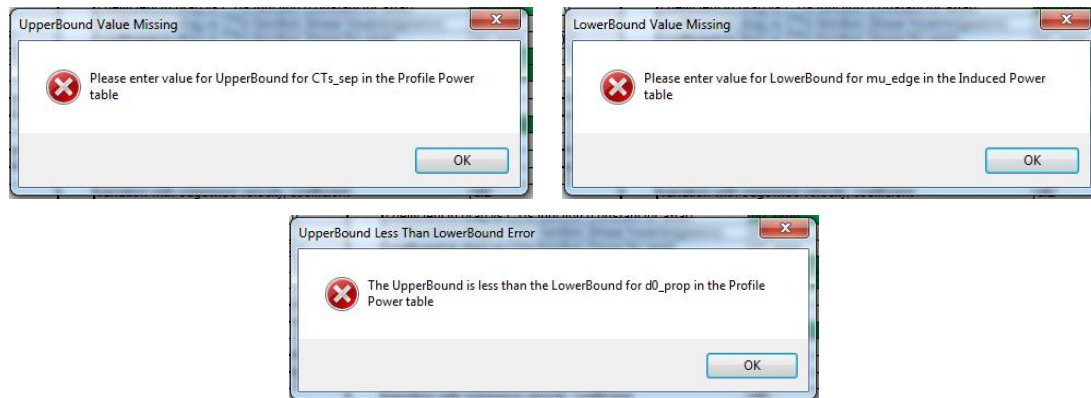
Figure 4-18: Possible error messages that occur when NDARC variables are not set correctly

4.5.3 NDARC – OPS Displays and Results

At the completion of the optimization run, a new sheet is generated in the NDARC – OPS under the "SaveAsFilename" name. An example of this sheet is shown in Figure 4-19. This sheet contains the NDARC variable values of the newly calibrated NDARC model, along with the complete calibration data set table with corresponding NDARC estimated values and relative residual errors for each calibration data point.
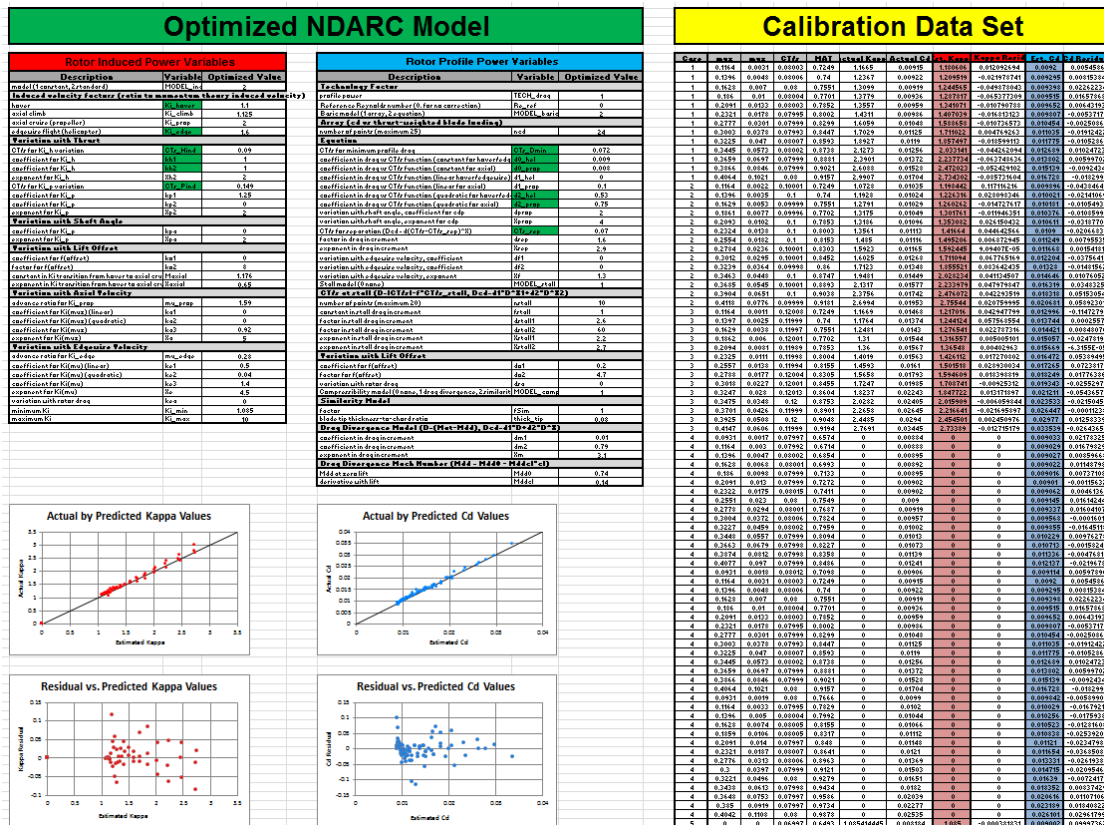
Figure 4-19: NDARC - OPS user display for results

To help visualize the accuracy of the results, an actual by predicted and residual by predicted plot are automatically generated for both the induced power coefficient and profile drag coefficient, shown in the bottom left of Figure 4-19. In the actual by predicted plot, a perfect fit is represented by the solid black line, which would indicate that for each calibration data point the estimated value from the NDARC curve fits perfectly matched the actual value from the comprehensive analysis data. The residual by predicted plot helps to visualize how large the residual is for each data point, where the residual for each data point is calculated as the relative percent error from the actual value. An example of a good fit (shown by the Kappa values) and a poor fit (shown by the Cd values) is provided in Figure 4-20. Though this example is exaggerated by stopping the profile drag coefficient optimization early, it still represents the trends in accuracy that should be looked for. That is, the closer the data points are to the "perfect fit" line and the smaller the residual errors, the more accurate the NDARC model fit.
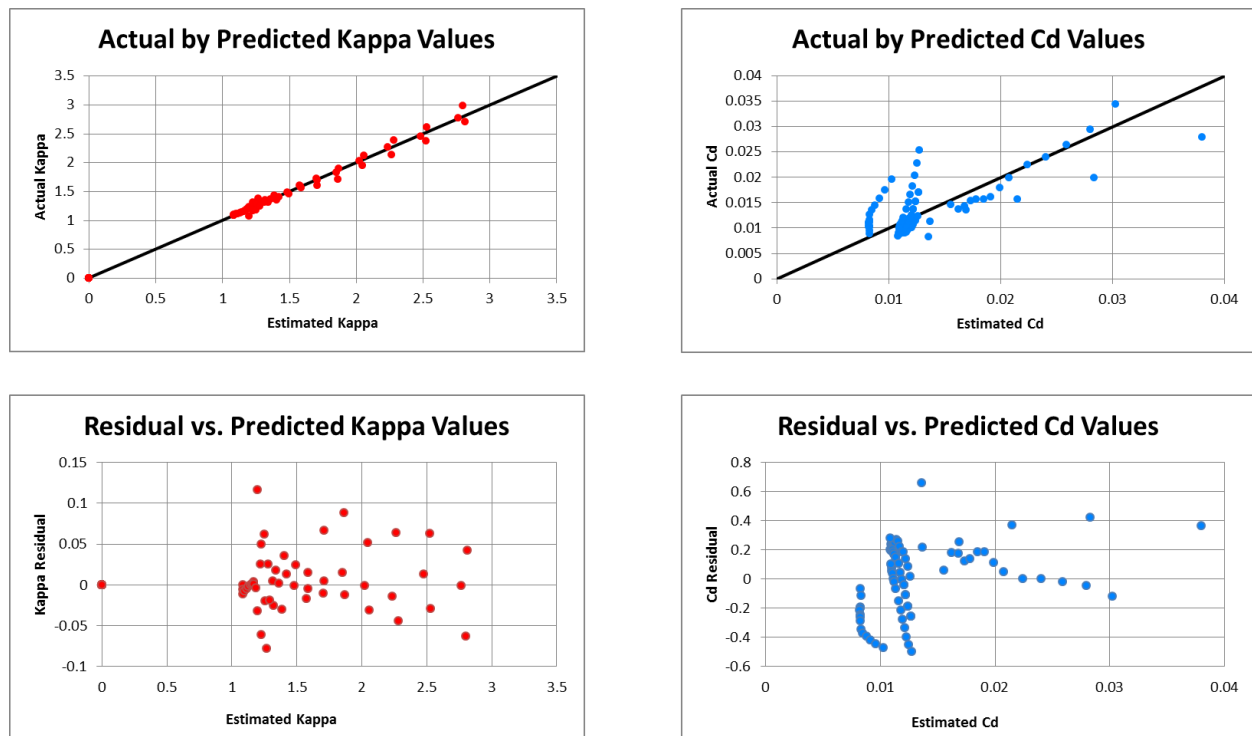
Figure 4-20: Example of results for a good fit (Kappa values) and poor fit (Cd values) on the actual by predicted and residual by predicted plots

4.6   NDARC – OPS Efficiency Case Study

A case study was performed to compare the required user effort and the accuracy of calibrating a NDARC model for rotor induced power using a manual calibration process versus the NDARC – OPS tool. To maintain consistency, the same NDARC variables used by the experienced user performing the manual calibration were used in the NDARC – OPS tool. The NDARC variables that were used as design variables

in this process are listed in Table 4-8, along with the lower and upper bound used in the NDARC – OPS tool. All other NDARC variables were set to default values (See Appendix G.2 for complete design space used). The run settings used for the NDARC – OPS calibration are shown in Table 4-9.

Table 4-8: Induced power design space used in NDARC - OPS for case study

Description	Variable	Lower Bound	Upper Bound	Resolution
Induced velocity factors (ratio to momentum theory induced velocity)				
hover	Ki_hover	1	1.3	0.001
Variation with Thrust				
CT/s for Ki_h variation	CTs_Hind	0	0.1	0.001
coefficient for Ki_h	kh1	-8	8	0.001
coefficient for Ki_h	kh2	-25	25	0.001
exponent for Ki_h	Xh2	0	4	0.001
CT/s for Ki_p variation	CTs_Pind	0	0.1	0.001
Variation with Edgewise Velocity				
advance ratio for Ki_edge	mu_edge	0	0.45	0.001
coefficient for Ki(mu) (linear)	ke1	-5	5	0.001
coefficient for Ki(mu) (quadratic)	ke2	-5	5	0.001
coefficient for Ki(mu)	ke3	-25	25	0.001
exponent for Ki(mu)	Xe	4	12	0.001

Table 4-9: Settings for NDARC - OPS calibration run

Number of Runs:	1	
Enter New Sheet Name Here:	NDARC OPS Run	
Induced Population/Generation Factor	4	8
Profile Population/Generation Factor	3	1

The case study used a calibration data set provided by Wayne Johnson for an unknown edgewise flight helicopter. This data set is provided in Appendix G.1. A summary of the results are listed in Table 4-10. The "user effort" time quoted in this table is the time required by the user to set up the calibration data set into a useable form, and then perform the calibration itself. For the NDARC – OPS tool, the run time of the optimization code (which was approximately 15 minutes on an i7 processor with 12 GB ram) is not included in the user effort, as the optimization code requires no user interaction or input at all, whereas the manual calibration requires the users input at all times.

Table 4-10: Results of NDARC calibration case study

Calibration Method	User Effort (min.)	Calibration Error
Manual Calibration	37 min.	3.98%
NDARC – OPS	6 min.	2.97%

The results overwhelming show that the NDARC – OPS tool is capable of obtaining more accurate results while requiring a fraction of the user effort. In addition, of the 6 minutes of user effort required for this case study, approximately five of those minutes were spent reformatting the calibration data provided into the table format required by the NDARC – OPS tool. Thus, if the calibration data is already provided in this table format, the user effort can be reduced to merely the time required to select the design variables and set the design variable ranges. Another comment that should be mentioned here is that the calibration error of the NDARC – OPS fell below that of the manual calibration error in less than ten seconds, but the code was allowed to run for the remainder of the 15 minutes to continue to explore the design space.

At this time, it is again stressed that the NDARC – OPS tool does not remove the need for SMEs input. Given the type of helicopter being modeled (i.e. edgewise flight versus axial flight), the SME must select the correct NDARC variables to optimize, while setting other NDARC variables to proper default values based on the current helicopter configuration. The NDARC variables that represent physical quantities (as discussed in Section 2593088.0.-946779886) must also be constrained to physically allowable values, or the models will have no physical significance. Given that a SME can properly set the design space for a given helicopter configuration, the NDARC – OPS becomes a valuable tool capable of rapidly calibrating NDARC models while requiring minimal user effort.

4.7   Conclusion

This report outlined the process used to develop the NDARC – Optimized Calibration (NDARC – OC) tool. The objective of the NDARC – OC tool is to automate the calibration of NDARC models against comprehensive analysis data. The optimization process is implemented in Python v3.4, using a genetic algorithm to perform the optimization. The optimization problem is set up through a distinct I/O file system, which allows the user to change run settings of the genetic algorithm, as well as change the design space with regards to what NDARC variables are design variables versus constant parameters.

For Windows OS users, an Excel GUI has been wrapped around the NDARC – OC tool to create the NDARC – Optimized Performance Spreadsheet (NDARC – OPS). The NDARC – OPS simply provides a GUI to allow users to set calibration data, as well as to define the design space and run settings for the optimization problem. A VBA script within the NDARC – OPS automates the process of writing the required input file for the NDARC – OC, reducing user error and effort required.

In Section 4.6 of this report, a case study is performed to compare the efficiency and performance of the NDARC – OPS against a previous method where the calibration is performed through manual iteration. The case study uses a set of calibration data provided by Wayne Johnson for an unknown edgewise flight helicopter (this data is provided in Appendix G). The manual calibration effort was performed by an experienced NDARC user, while a second model was calibrated with the NDARC-OPS tool using the same NDARC variables as the expert used. The results showed that the NDARC – OPS tool was capable of getting more accurate results than the manual calibration at a greatly reduced effort to the user.

There are several areas that could be investigated to improve the performance of the NDARC – OC tool. Alternative optimization algorithms could be implemented if it is believed that they will outperform the genetic algorithm implemented in efficiency and/or consistency of results. If a new optimization algorithm is being investigated, the concerns noted in Section 4.3.2 should be taken into consideration. Additionally, many of the limitations of the NDARC – OC code revolve around computational limitations that arise when the design space grows exponentially. To partially address this issue, the code could be converted to a faster programming language such as Fortran or C/C++. However, the Python version of the code uses the numerical python (NumPy) library whenever possible, which is a pre-compiled C code for efficiency. Thus, the speed-up obtained by switching to Fortran or C/C++ may not be as large as one might expect.

5. Framework for linking system capability uncertainty to individual technologies and groups of technologies in a portfolio: uncertainty ovals –or- uncertainty around tech factors

5.1 Introduction

The following section documents the work done in improving vehicle performance analysis methods including quantification of uncertainty. The impact of technologies as well as the impact of variance are studied and presented. To quantify uncertainty, first, the sources of uncertainty in performance analysis must be established. Once these sources are identified, their effects are quantified by using the Probabilistic Certificate of Correctness (PCC) methodology; in which simulations are performed to establish confidence in predicted performance.

Uncertainty assessment of complex systems has been studied for many projects, and an important body of literature is available in the subject. First, a literature review of the uncertainty sources throughout the vehicle lifecycle is shown in Figure 5-1. [22]
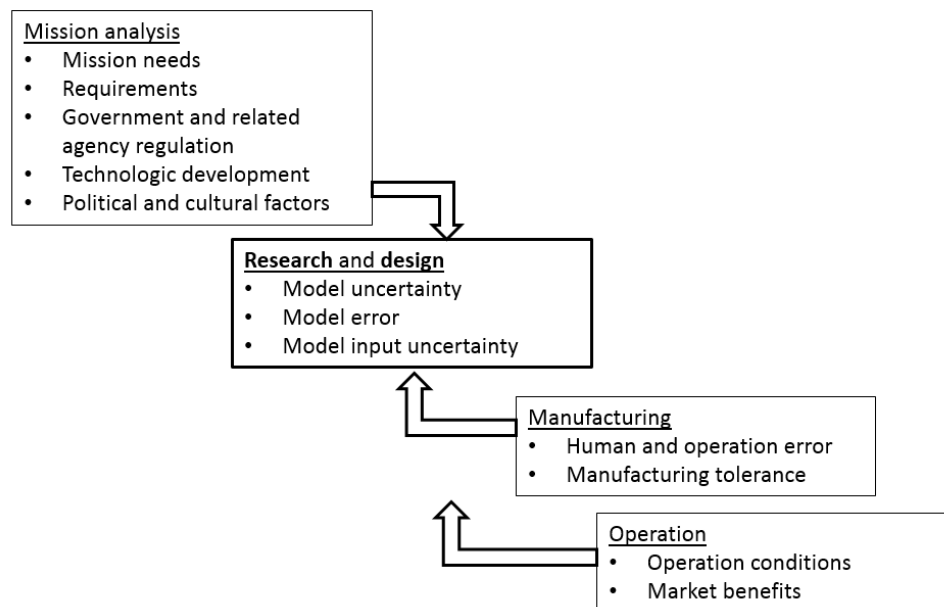
Figure 5-1. Uncertainty Sources Throughout the Vehicle Lifecycle

5.2 Sources of Uncertainty

Previous literature [22] also outlined the sources of uncertainty.
- Physical and mathematical modeling
  - Ignorance

91

- o Lack of understanding
- o Incomplete knowledge
- Model uncertainty
  - o Inherent system variations
  - o Model structure uncertainty
  - o Model Parameter uncertainty
- Model error
  - o Computational implementation and numerical programming
  - o Discretization, round off, programming error
- Computational simulation model
  - o Model uncertainty
  - o Model error
- Model input uncertainty
  - o Design variable uncertainty

5.3 Uncertainty in numerical simulations
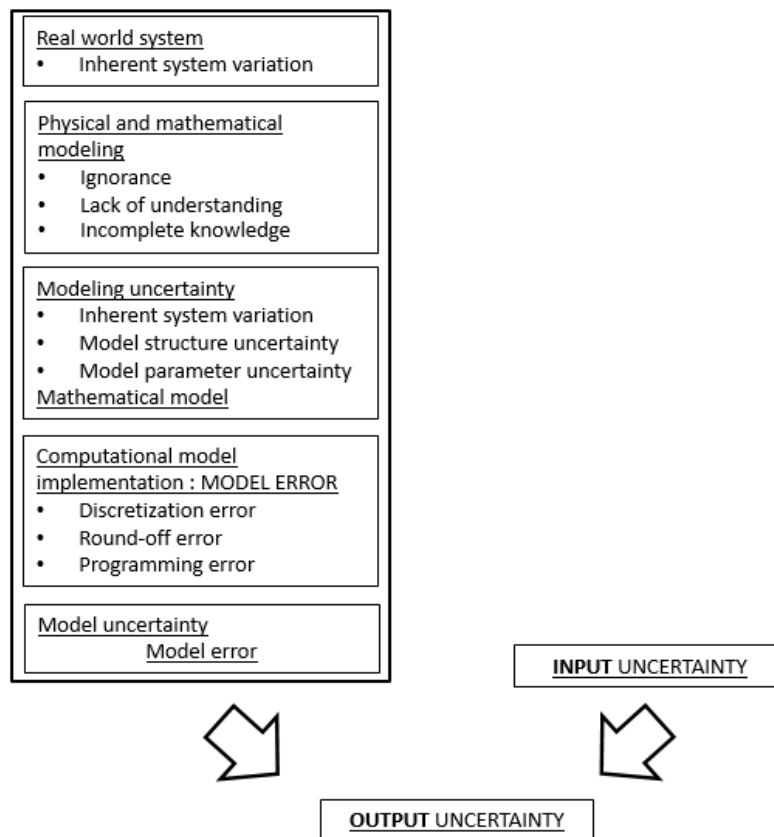
Figure 5-2. Uncertainty in Numerical Simulations [22]

5.4   Uncertainty in CATE

The current Uncertainty module in CATE shown in Figure 5-3 allows for a Monte Carlo simulation of the sizing output subjected to user defined uncertainty on the sizing parameters. The uncertainty on the input can be a uniform distribution or a normal distribution. This process is done on the sizing of the aircraft. Consequently, the input are related to the mission parameters, the sizing condition and technology factors. The output is expressed graphically and is related to vehicle weights, size (geometrically and power), and weights.
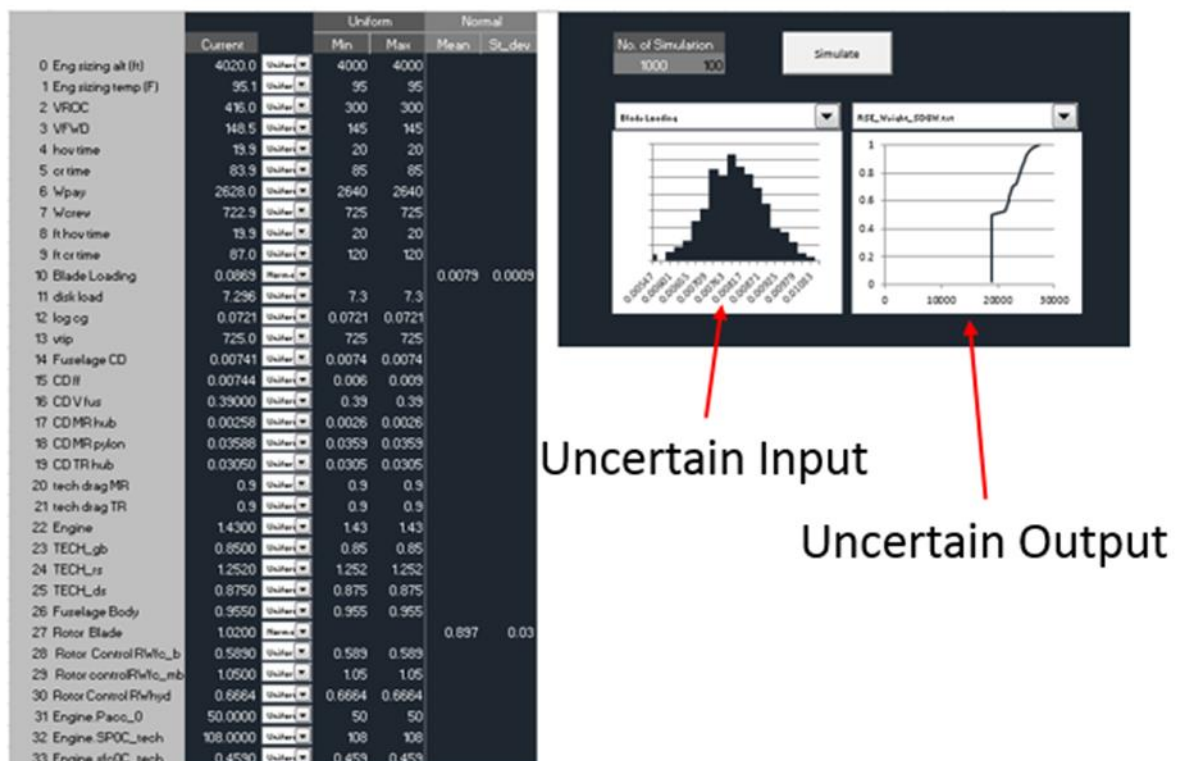
Figure 5-3. Current CATE Uncertainty Dashboard

5.5   PCC Methodology

The Probability Certificate of Correctness (PCC) methodology has been studied for rotorcraft. The process is outlined below and shown pictorially in Figure 5-4. It allows to perform a probabilistic assessment of the performance goals based on a numerical process.

- 1- Procedure:
  - o   Error distribution estimation
  - o   Assign distribution to appropriate variables
  - o   Run Monte Carlo Simulation (MCS)

o   Derive performance metric probability density function

- 2- Technology impact (UH-60M)

    o   Distribution on technology impact factors

        ▪   Rotor aerodynamics, engine performance, and tail drag

    o   Probabilistic assessment of performance goals
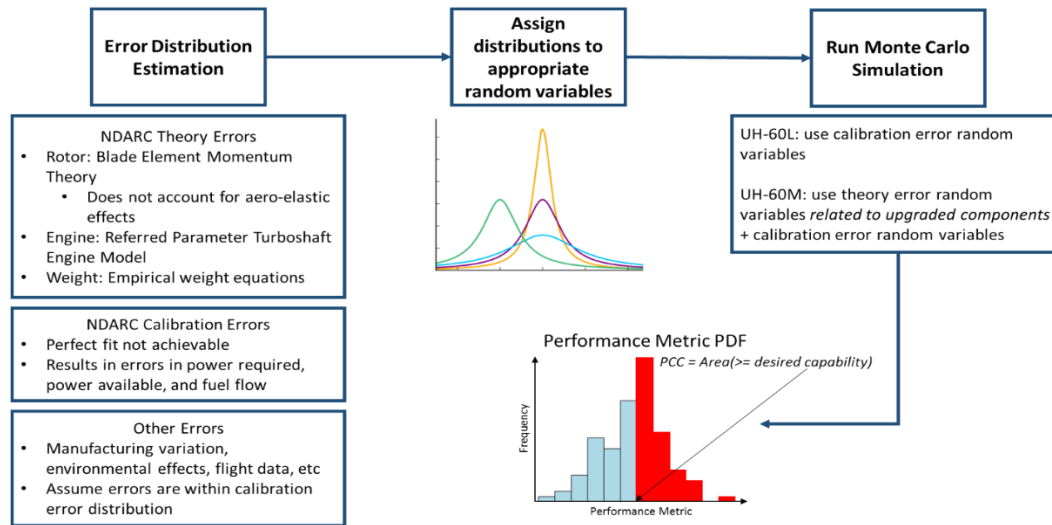
Figure 5-4. PCC Methodology

5.6   Uncertainty in NDARC

The vehicle design and analysis in CATE has been performed through NDARC. This software has its own sources of uncertainty. Because NDARC is component based, the breakdown of the sources of uncertainty was performed accordingly.

- Rotor performance:

The rotor model, as suggested by Johnson, is based on a decomposition of the profile power and induced power. The process is to take flight test data, perform a higher fidelity rotor analysis of the data using RCAS or CAMRAD and extract a set of parameters describing the profile and induced power coefficient as a function of various parameters. This last step is done using *the rotor spreadsheet*, a spreadsheet that helps the tuning of those parameters. Consequently, the sources of error can come from:

- Flight test : Test conditions (ex: trim)

- High fidelity rotor analysis: error associated with the numerical representation in the analysis tool

- Rotor spreadsheet: error on the calibration of the parameters

94

- Engine performance

Unfortunately, there is no knowledge on engine performance error modeling from NDARC documentation as engine decks often contain proprietary data. In the past, the CATE team calibrated the power available and fuel flow with user manual available data. In this case, error can come from

- Error on available data (form the user manual itself)
- Error on the calibration method
- Weights

The weight models in NDARC have published error models associated with each one of them.

5.6.1 Bayesian approach to the rotor spreadsheet calibration

As mentioned previously, part of the rotor calibration is to find the parameters that represent the coefficient of induced and profile power as a function of the operating condition. Given the models and available performance data, Bayesian statistics can be used to find calibration related uncertainty.

In a test case, the 3 variables representing the Cd Mean as a function of CT/sigma in the rotor spreadsheet were calibrated using Bayesian statistics. Note that only data at low CT/sigma was used (before any stall on the rotor). For low Ct/sigma, the model is a quadratic. The equation is:

$$Cd_{mean} = a_1 + a_2 \frac{CT}{s} + a_3 \left(\frac{CT}{s}\right)^2$$

Figure 5-5 illustrates data coming from higher fidelity tool for 2 altitude conditions. The blue dots are data, and the orange line is the least-square fit of the parameters a1 a2 and a3. It can be seen that a least-square cannot be fit perfectly with the second order model. Consequently, there is uncertainty in the representation of the model through this model.
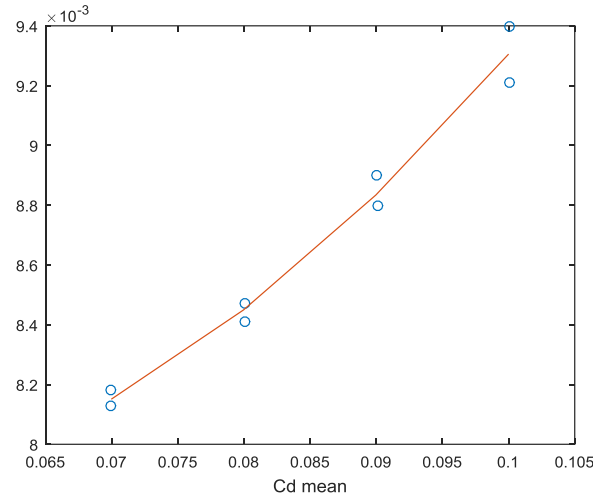
Figure 5-5. Cd Mean Data and the Exponential Curve Model

A Bayesian approximation of the 3 parameters describing the second order Cd_mean model was performed and is shown in Figure 5-6. A prior distribution (blue curves) was associated with each parameter, based on the least-square regression performed before and engineering judgment. The data was used to update the Bayesian model and the posterior distribution of each parameter is an output expressed by the orange curves. This representation allows to associate a distribution to each coefficient used in the numerical model. This can be used to performed uncertainty analysis.
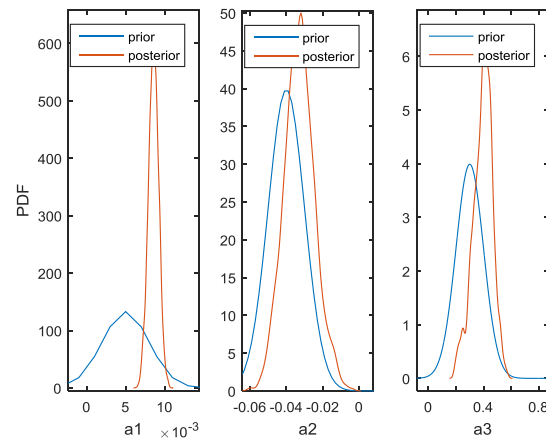
Figure 5-6. Bayesian Approx. of the 3 Parameters Describing Second Order Cd Mean Model

5.6.2   Uncertainty propagation methods

This section evaluates how to propagate the uncertainty of the max speed of a helicopter based on uncertainty on the operating condition, using either Taylor series approximation or Monte Carlo simulation.

The performance characteristic studied was the max speed of a UH60-L evaluated in through a performance calculation in NDARC. The input that were assumed as uncertain were the CG location, the operating temperature and the altitude conditions.

In order to propagate uncertainty, two methods were evaluated: 1) The Monte Carlo Simulation and the 2) Taylor Series Approximation.

The Taylor series approximation assumes a linear variation of each output/input pairing. The variance and average of the input are propagate to the output by the equation below. [22]

$$\mu = E(x), \ \sigma = \sqrt{\left(\Sigma \frac{\partial f}{\partial x_i}\right)^2 \sigma_{x_i}^{\ 2}}$$

A more involved formulation allows one to take into account input coupling and co variance. The only modeling required is the local derivative (1 variable) or the local derivatives of each output to each (Jacobian).

A sweep of CG location was performed, as is shown in Figure 5-7. This results shows that the speed as a function of the Cg Location cannot be treated linearly. Consequently the Taylor Series approximation cannot be used, and a Monte Carlo Simulation was performed. The Monte Carlo simulation is more computationally expansive, as the input distribution are approximated and ran multiple times and the output distribution can be assessed.

The inputs are shown in Table 5-1 and results are shown in Figure 5-8. The distribution on the CG location was based on the uncertainty of the CG location of a CH47 [23]
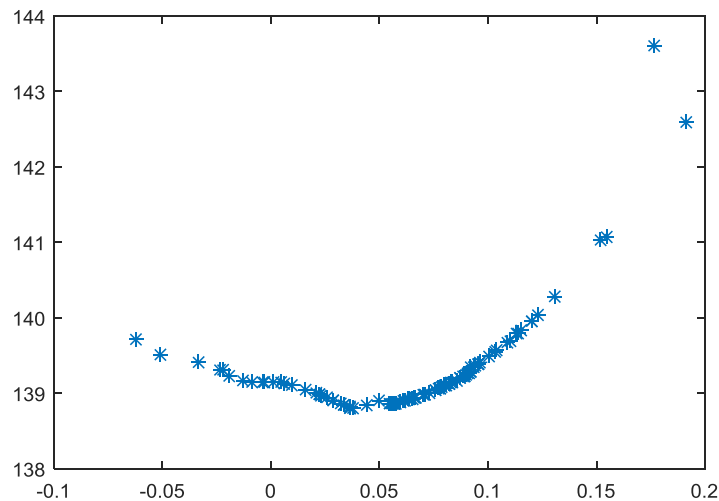
Figure 5-7. Maximum Speed vs. Center of Gravity Location

Table 5-1. Inputs to the Monte Carlo Simulation

Variable	Mean	Sigma
CG	0.07	0.05
Temp	95F	5F
Alt	4000'	300'
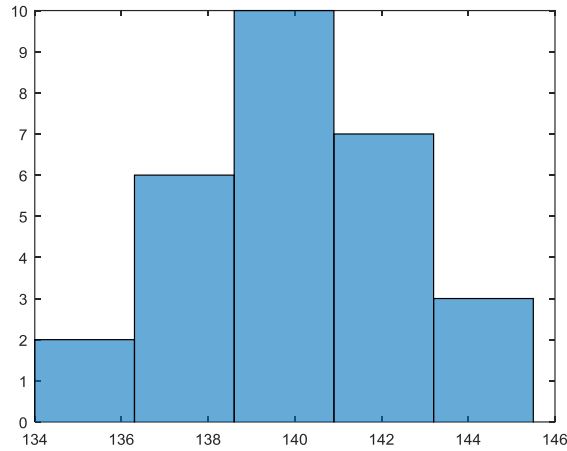
Figure 5-8. Maximum Speed Distribution, Standard Deviation of 2.4 kts

The standard distribution of the maximum velocity is 2.4knots. This can be used as a first approximation of the uncertainty during the flight test for example, and can be used to inform the uncertainty about the calibration. Due to lack of interest in the uncertainty in the calibration of the model, the task was discontinued.

5.7 Conclusion

Uncertainty on the design and performance of rotorcraft can come from various sources. Some aspects of the CATE process were studied, and specific examples were used as test cases to demonstrate possible notional process of uncertainty assessment and propagation, specifically in the calibration process.

6. UH-60 Upgrade Study

6.1 Introduction

Using the CATE integration environment, two possible UH-60 upgrade studies (5-blade rotor system, ITEP engine) have been performed. The detailed descriptions of the investigation methodologies and the results are summarized in the following chapters.

6.2 5-Blade Rotor System Investigation

To demonstrate the procedure from a high fidelity analysis tool (RCAS) to the CATE (NDARC) environment through the calibration step, a 5-blade rotor system has been investigated. As the first step in the investigation, the UH-60A NDARC model has been updated to match with the published data better. The new UH-60A model results have been obtained by modifying the engine parameters and weight factors and compared with the published data and the previous results in the Table 6-1. As shown in the table, new calibration results show more close to the published data.

Table 6-1. New Calibration Results

Sizing Results	Published	Previous Calibration	New Calibration	Diff (%)
Design Gross Weight (lbs)	16,500	17,088.8	**16,493.5**	0.04
Drive System Limit (HP)	2,828	2,805.1	**2,828.2**	0.007
IRP Power SLS (HP)	1,560	1,762.9	**1,560.7**	0.04
MCP Power SLS (HP)	1,313	1,487.6	**1,317.0**	0.3

For the 5-blade rotor system analysis, either fixing the solidity by the reduced blade dimensions or increasing the solidity by adding a same blade can be possible. In case of the same solidity analysis, there is no difference in the performance results by the RCAS analysis, so the case with the increased solidity analysis with the additional same blade has been chosen and performed. If this investigation is aimed at an optimized design analysis, then an intensive parametric investigation and optimization among the solidity, the rotor speed, and the blade configuration parameters need to be performed, along with structural and dynamic analyses. However, the current investigation is focused on linking the RCAS and the CATE (NDARC) environment and detailed parametric optimization is beyond the work scope, only the parametric investigation has been performed and the analysis results below should not be considered as the optimum design results for the 5-blade rotor system.

Figure 6-1 and Figure 6-2 show the hover rotor power comparison at the sea level standard condition and the 4,000ft, 95F condition. As shown in these figures, the power required for the 5-blade rotor system is a little higher in the UH-60 C_T range (~0.007) because the current 4-blade rotor system is optimized for the UH-60 weight range. However, as the C_T values increase, the 5-blade rotor system becomes more efficient and results in lower power required due to the induced power reduction.
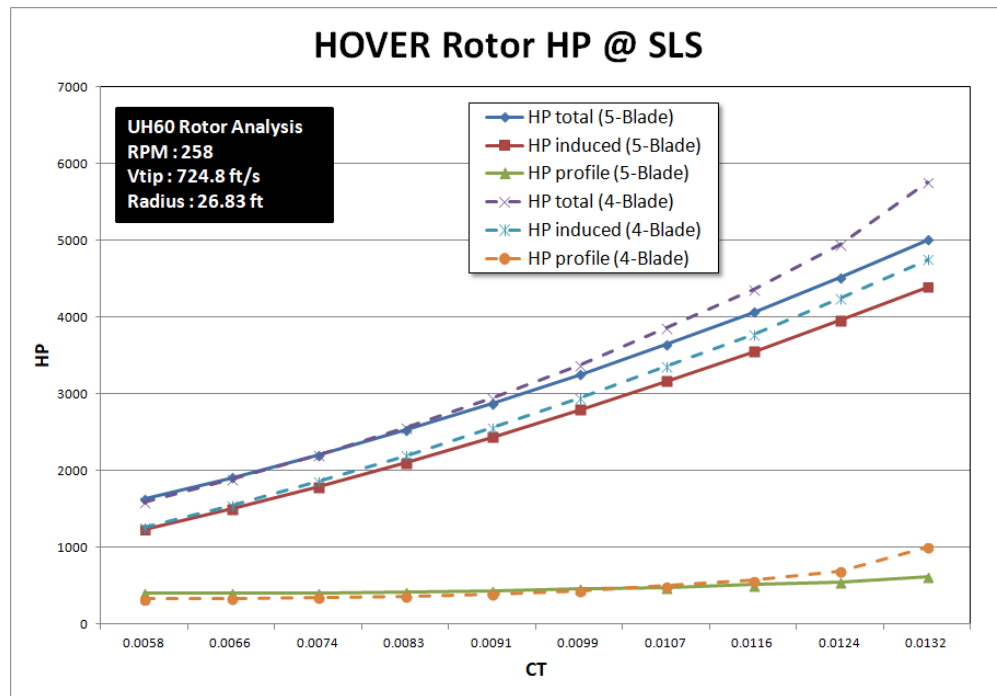
Figure 6-1. 4-Blade/5-Blade Hover Power Comparison at SLS Condition
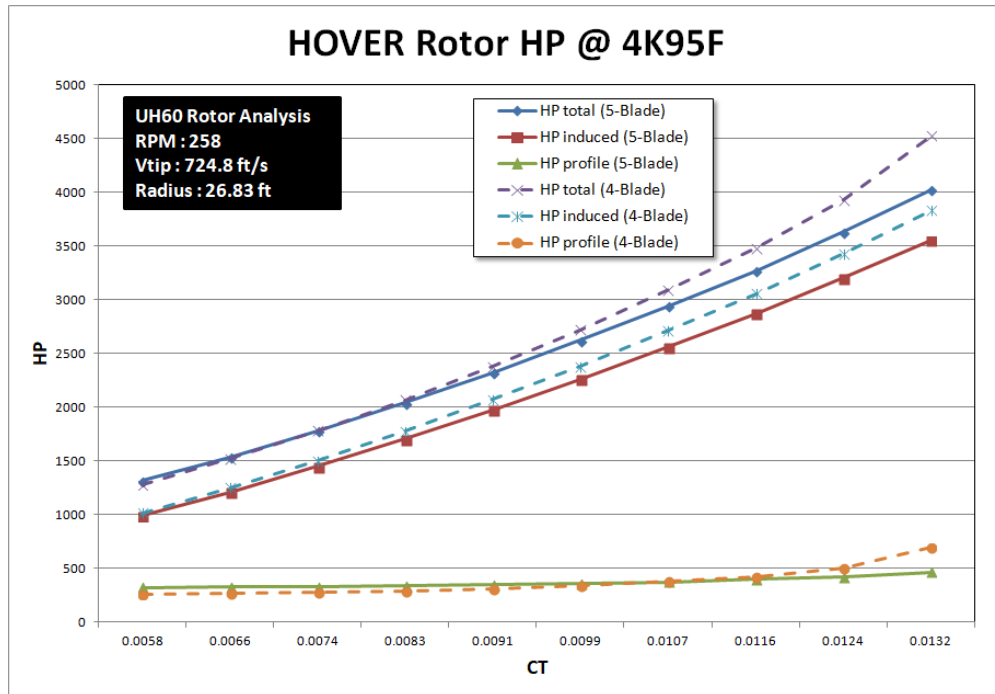
Figure 6-2. 4-Blade/5-Blade Hover Power Comparison at 4K/95F Condition

Figure 6-3 and Figure 6-4 show the forward flight power required at two different C_T conditions. The C_T values are close to the UH-60 value for both cases, so there show higher power required for the 5-blade rotor system. But the difference becomes smaller at the higher C_T case (Figure 6-4), which is the same trend with the hover results. This trend shows more clear in the Figure 6-5 which includes three different C_T conditions. It is obvious that the 5-blade rotor system without reducing the rotor speed should be less efficient in the UH-60 C_T range.

Thus, more analyses with the different RPM values have been conducted with the C_T value fixed as 0.0083 and the results are shown in the Figure 6-6. With the RPM reduced, the rotor system is operated within the more efficient rotor pitch angle ranges, so the power required gets reduced. However, this rotor speed investigation should be conducted with the dynamic stability analysis and the optimum RPM can be found with a more comprehensive analysis.
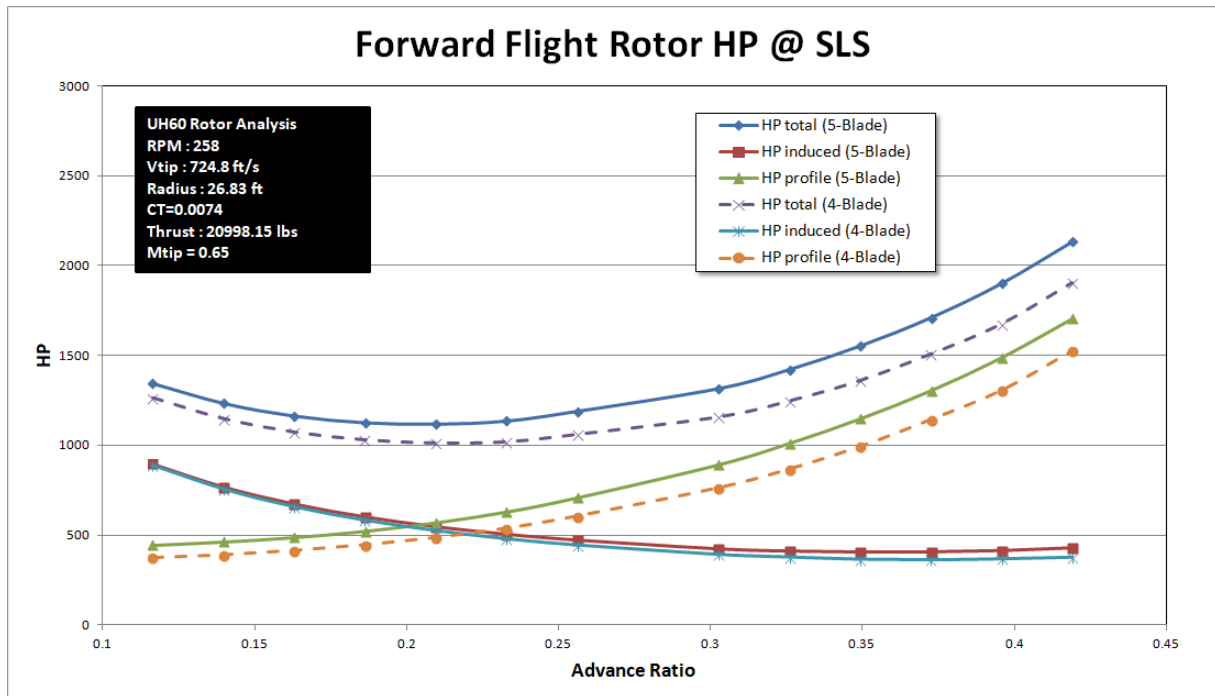
Figure 6-3. 4-Blade/5-Blade Forward Flight Power Comparison ($C_T = 0.0074$)
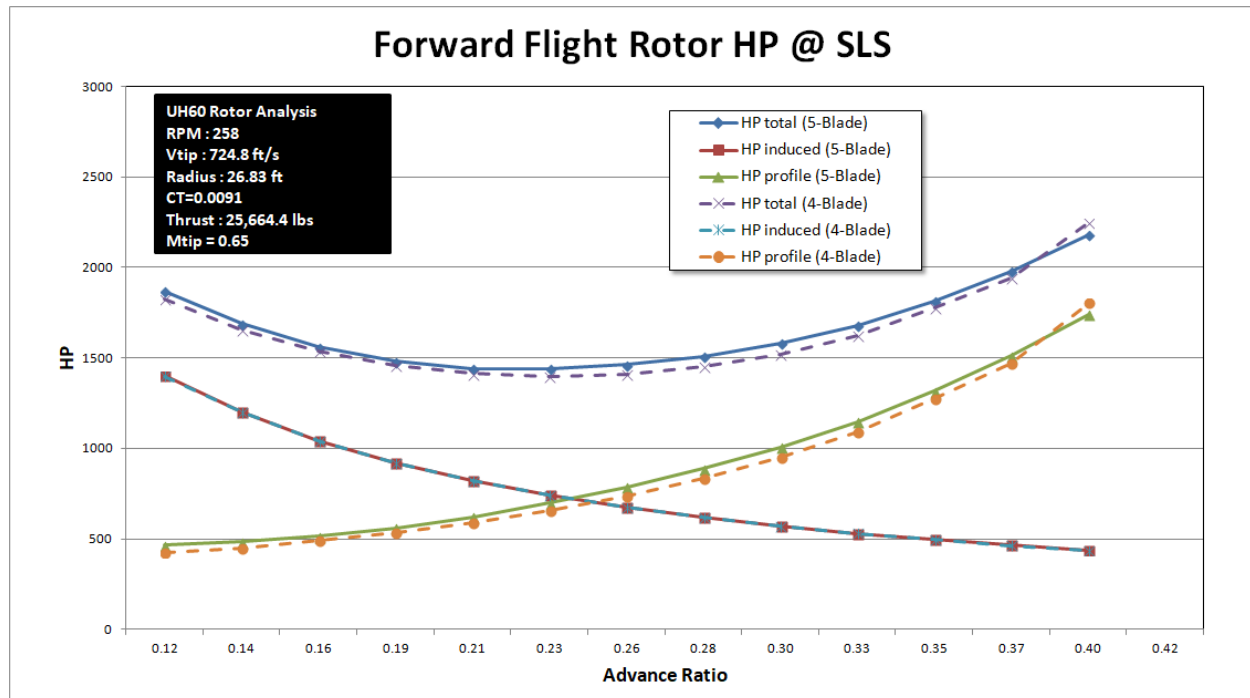
Figure 6-4. 4-Blade/5-Blade Forward Flight Power Comparison ($C_T=0.0091$)
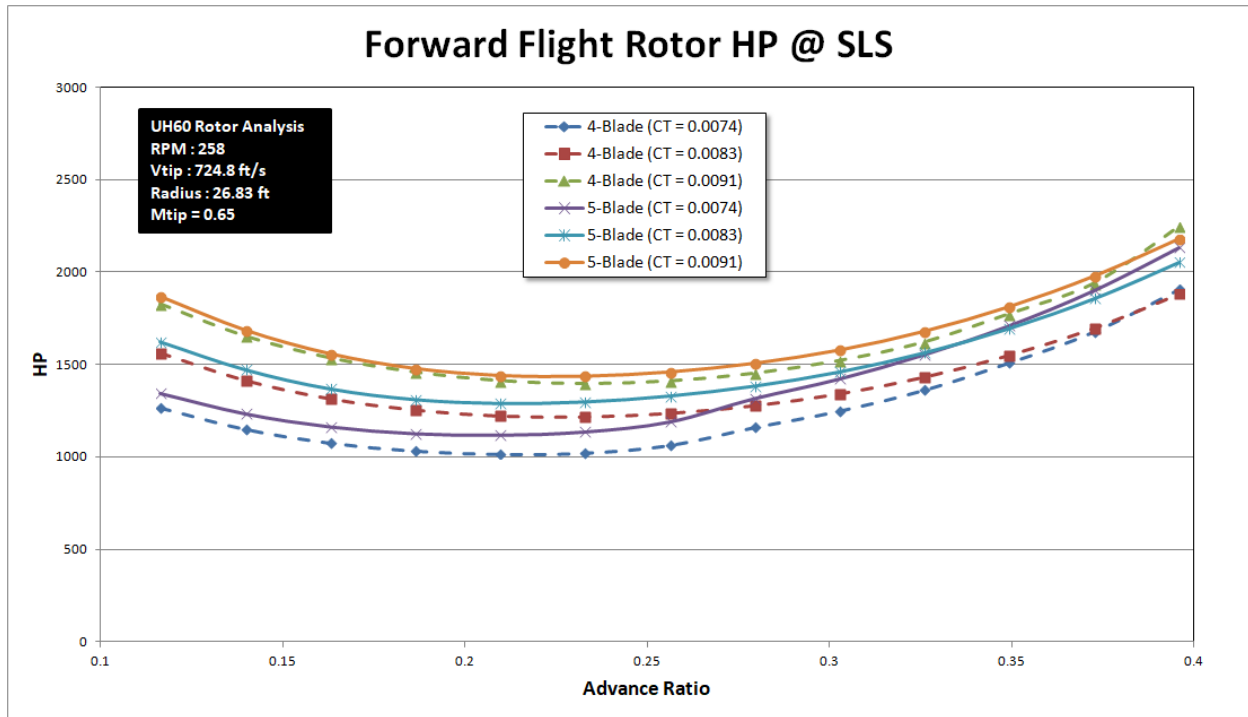
Figure 6-5. 4-Blade/5-Blade Forward Flight Power Comparison (RPM=258)
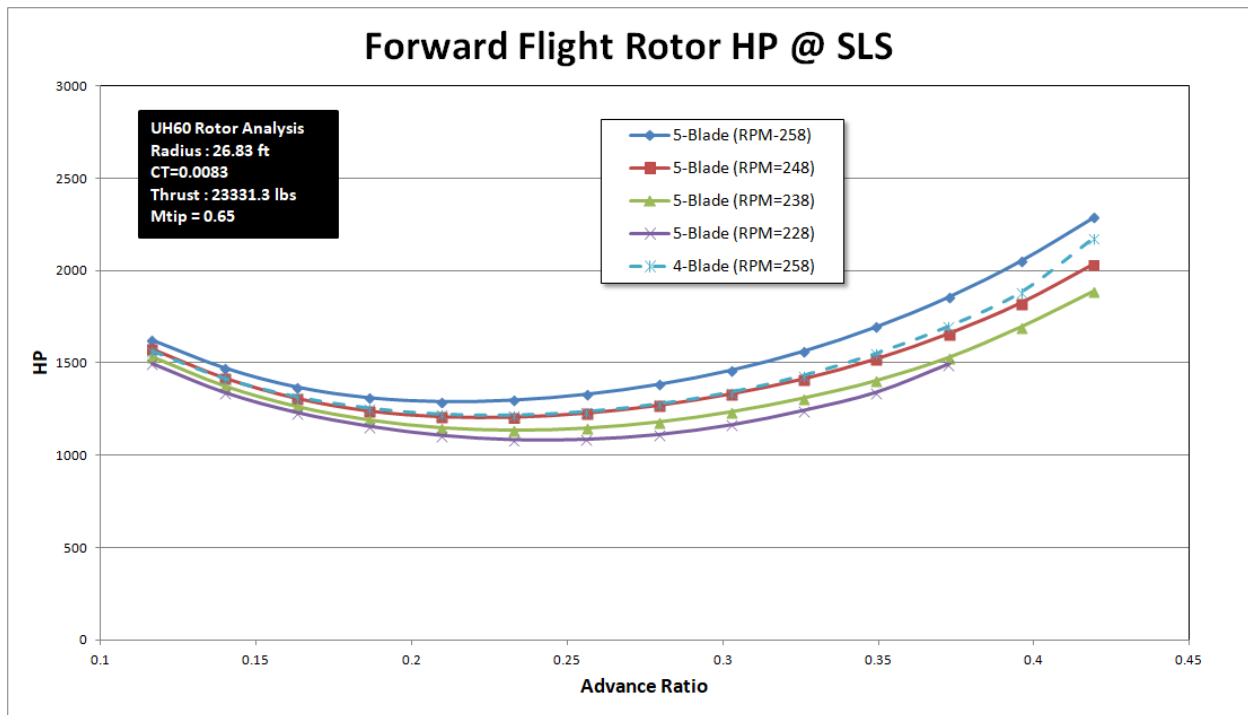
Figure 6-6. 4-Blade/5-Blade Forward Flight Power Comparison (C_T=0.0083)

To connect the RCAS results into the CATE (NDARC) environment, the Optimized Performance Spreadsheet (OPS) analysis has been performed as shown in the Figure 6-7.
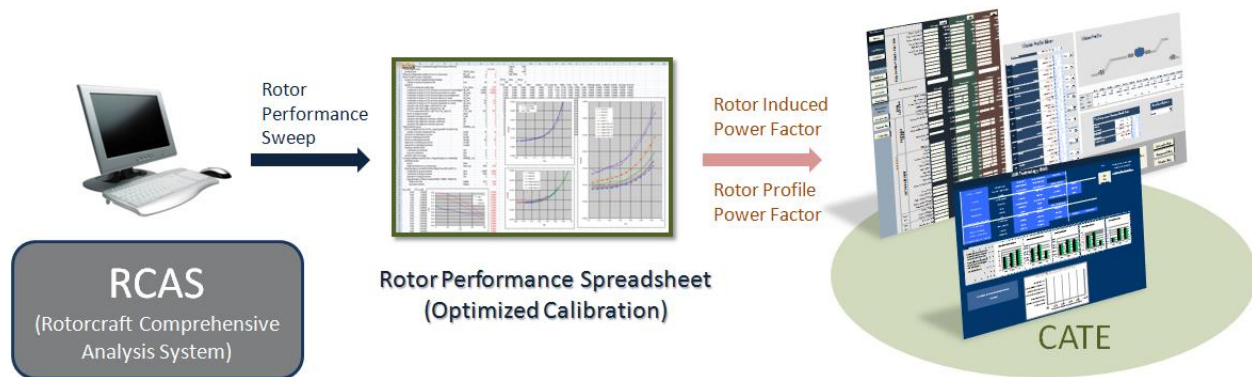
Figure 6-7. Integration Flow between the RCAS and the CATE

The hover induced power related NDARC variables and the profile power related variables have been obtained using the spreadsheet and the resulting hover induced power factor and the mean drag coefficient are plotted in comparison with the optimized spreadsheet results in the Figure 6-8, which show good correlation.
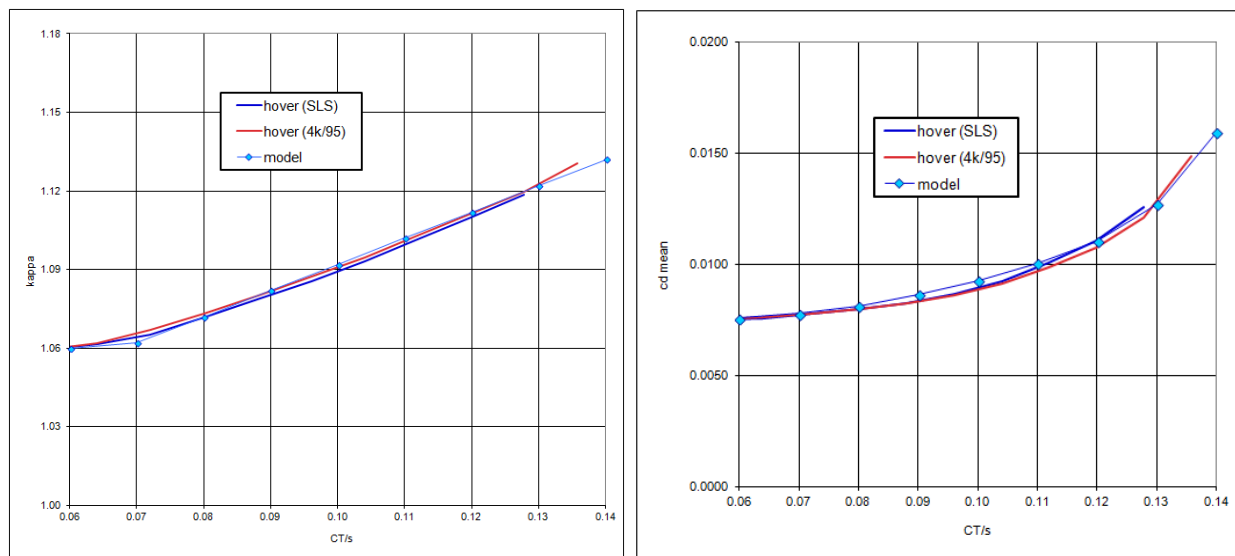
Figure 6-8. Hover Induced/Profile Parameters Comparison

RCAS forward flight induced power trend is different from the optimized spreadsheet trend in the high speed region, so the forward flight rotor power factors are manually tuned to match with the total power required.

Table 6-2 and Table 6-3 include the list of the NDARC variables identified through this procedure.

Table 6-2. Rotor Induced Power NDARC Variables

Description	Variable	Value
model (1 constant, 2 standard)	MODEL_ind	2
Induced velocity factors (ratio to momentum theory induced velocity)		
Hover	Ki_hover	1.086
Axial climb	Ki_climb	1.11
Axial cruise (propeller)	Ki_prop	2
Edgewise flight (helicopter)	Ki_edge	2
Variation with Thrust		
CT/s for Ki_h variation	CTs_Hind	0.09
Coefficient for Ki_h	kh1	0
Coefficient for Ki_h	kh2	0
Exponent for Ki_h	Xh2	2
CT/s for Ki_p variation	CTs_Pind	0.1
Coefficient for Ki_p	kp1	1.25
Coefficient for Ki_p	kp2	0
Exponent for Ki_p	Xp2	2
Variation with Edgewise Velocity		
Advance ratio for Ki_edge	mu_edge	0.28
Coefficient for Ki(mu) (linear)	ke1	0.51
Coefficient for Ki(mu) (quadratic)	ke2	0.03
Coefficient for Ki(mu)	ke3	1
Exponent for Ki(mu)	Xe	4.56
Variation with rotor drag	kea	0
Minimum Ki	Ki_min	1.06
Maximum Ki	Ki_max	10

Table 6-3. Rotor Profile Power NDARC Variables

Description	Variable	Value

Technology Factor		
Profile power	TECH_drag	1
Reference Reynolds number (0. for no correction)	Re_ref	0
Basic model (1 array, 2 equation)	MODEL_basic	2
Array (cd vs thrust-weighted blade loading)		
Number of points (maximum 25)	ncd	24
Equation		
CT/s for minimum profile drag	CTs_Dmin	0.05
Coefficient in drag vs CT/s function (constant for hover/edgewise)	d0_hel	0.0075
Coefficient in drag vs CT/s function (constant for axial)	d0_prop	0.0083
Coefficient in drag vs CT/s function (linear hover/edgewise)	d1_hel	0
Coefficient in drag vs CT/s function (linear for axial)	d1_prop	0
Coefficient in drag vs CT/s function (quadratic for hover/edgewise)	d2_hel	0.7
Coefficient in drag vs CT/s function (quadratic for axial)	d2_prop	0.5
CT/s for separation (Dcd = d(CT/s-CT/s_sep)^X)	CTs_sep	0.07
Factor in drag increment	dsep	4
Exponent in drag increment	Xsep	3
Variation with edgewise velocity, coefficient	df1	0
Variation with edgewise velocity, coefficient	df2	0
Variation with edgewise velocity, exponent	Xf	2
Stall model (0 none)	MODEL_stall	1

The Sizing results comparison between the UH-60A and the 5-blade rotor configuration with the reduced RPM of 238 has been performed and the results are shown in the Table 6-4. When being sized based on the same design mission, the 5-blade rotor configuration shows slightly higher design gross weight because the C_T value is low around in a 0.006 range and the empty weight increases due to the additional blade.

Table 6-4. Sizing Comparisons of 4-blade and 5-blade rotor system

Sizing Results	4-Blade UH-60A	5-Blade UH-60A
Design Gross Weight (lbs)	16,493.5	16,565.4
Empty Weight (lbs)	11,026.6	11,131.0

Table 6-5 shows the performance run results without the sizing run at the 22,000 lbs MTOW condition. The vertical rate of climb doesn't change because the rotor power required at the 22,000 lbs ($C_T = 0.0078$) is almost same for the 4-blade system and the 5-blade rotor system as shown in the Figure 6-1 and the

maximum speed improves slightly. Thus, as shown in these results, the 5-blade rotor system investigation doesn't show significant performance improvements. This conclusion is possibly due to the lack of the comprehensive optimization analyses, and the better design results can be obtained if additional higher fidelity analyses tools are connected to the CATE environment and the integrated procedures and analyses are performed.

Table 6-5. Performance Comparisons of 4-blade and 5-blade rotor system

Performance Results	4-Blade UH-60L	5-Blade UH-60L
MTOW (lbs)	22,000	22,000
VROC (ft/min)	726	726
Max Speed (kts)	147	150

6.3 UH-60 with the ITEP Engine

An upgrade of the UH-60 from the baseline engine to General Electric's ITEP (T901 Turboshaft) engine has been investigated. The ITEP engine represents a new technology that, when combined to the existing UH60 airframe, is cited by GE to yield the following performance improvements:

- 50% more power at SL/ISA
- 40% more power at 4k/95
- 25% reduced fuel consumption
- Lower maintenance costs

Evaluation of this upgrade has been conducted through trade studies outlined in the environment below and compared to the existing UH60 performance outlined in NASA's NDARC analysis tool.

6.3.1 Trade Study Environment

The current trade study involves collecting a series of results from NDARC for comparison. To accelerate the process, an environment was prepared in ModelCenter which can be seen in Figure 6-9. A QuickWrap model parses information from the NDARC inputs, runs NDARC, and reads the output files. That information is shared between Excel and Matlab. Excel stores the values for record while Matlab does the calculations and produces graphs of the power sweeps for sea level standard and high, hot day.
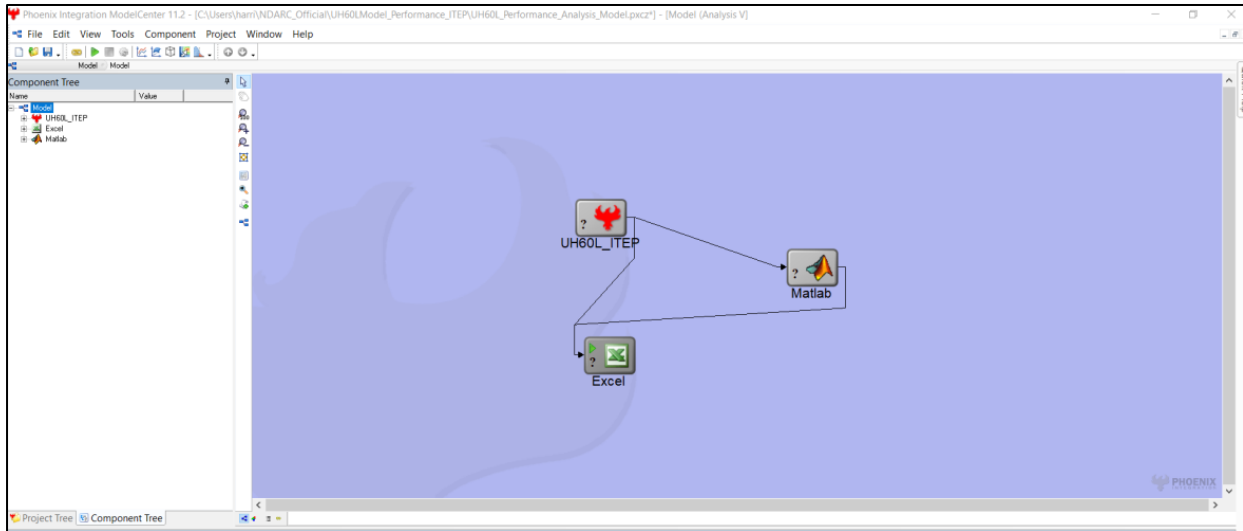
Figure 6-9. Trade Study Environment Built in ModelCenter

6.3.2 Analysis Methodology

The capability of the UH60 with ITEP upgrades was assessed by comparing the power sweep curves of the rotorcraft in forward flight using each engine to obtain crucial changes in performance such as hover rate of climb margin, maximum forward flight speed, and the maximum power consumed. This was accomplished using NASA's NDARC integrated into the trade study environment described above.

Another critical aspect to consider in evaluating the upgrade to ITEP is the increased robustness offered by a more powerful engine against performance losses or compromises on other systems of the rotorcraft. For example, a more powerful engine may offer equivalent top speeds to the baseline except at higher parasitic drag coefficients, allowing for more external storage for the same performance as the baseline. Evaluations of these sensitivities/robustness were conducted for the following cases using the integrated trade study environment with NDARC.

Table 6-6. Metrics evaluated for upgrade sensitivity and robustness

Hover Rate of Climb Margin
Parasitic Drag Coefficient
Fuel Weight Consumed
Range
Endurance
Maximum Speed
Endurance at Hover Ceiling

6.3.3 Comparison Results

Using NASA's NDARC, the following results were generated for a forward flight power sweep using the UH60's baseline engine and the higher performance ITEP engine under consideration for FVL. Initially, the UH60L was sized with and without the ITEP engine to determine the predicted effects of the ITEP engine. As seen in Table 6-7. , the empty weight does increase from the size of the ITEP engine as compared to the sized rubber engine. Table 6-7 also shows that the fuel required decreases although. This is because of the increased efficiency of the engine. Overall, the increase in design gross weight is negligible and it can be claimed that the sizing process for the specific missions in consideration produces the same design gross weight. Thus, the performance analysis has been be done with each vehicle at max takeoff weight (MTOW) for the UH60L, which is 22,000 lb.

Table 6-7. ITEP vs. non-ITEP Sized Vehicle Comparison

Variable	Units	ITEP % Change
Weight Empty	lb	3.42%
Fuel Weight	lb	-9.03%
DESIGN GROSS WEIGHT	lb	1.34%
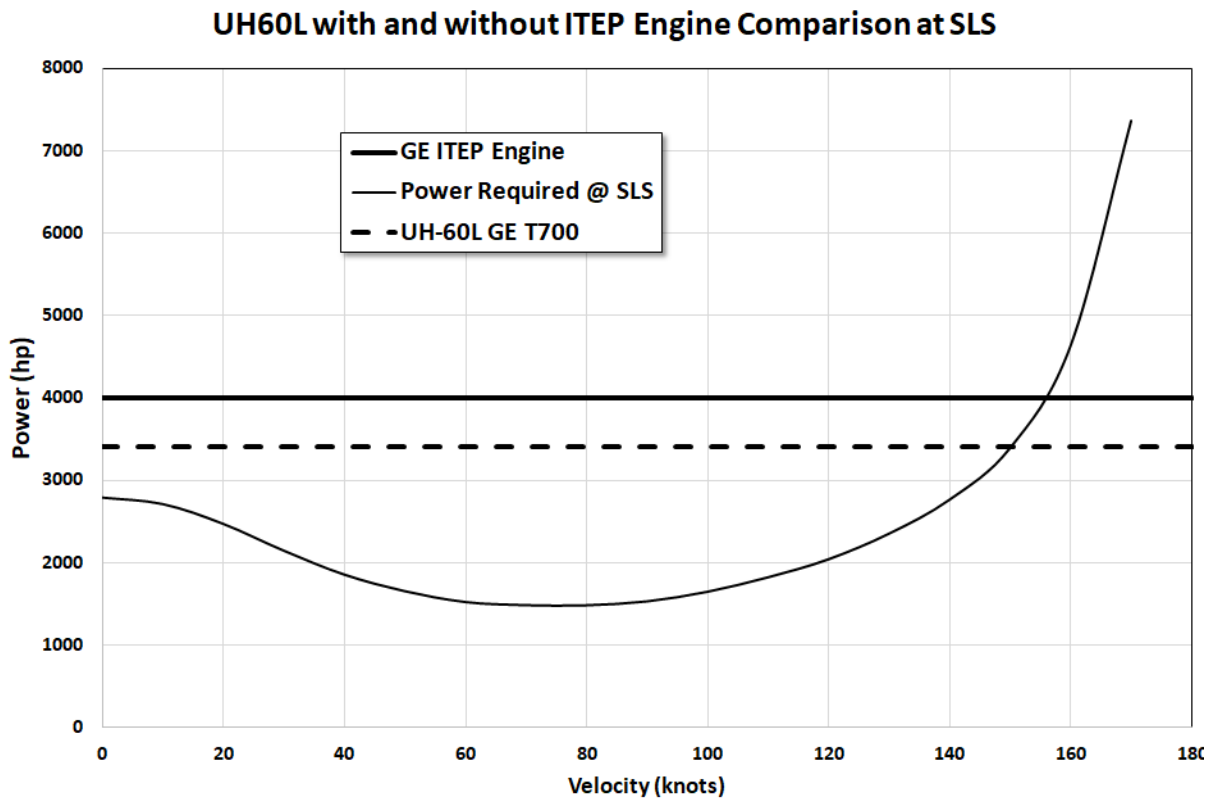
Figure 6-10. Power Required and Available Comparison with/without ITEP Engine

The comparison of the forward flight power required and power available at SLS condition is shown in the Figure 6-10. From the chart, the vertical rate of climb and the maximum speed are obtained and summarized in the Table 6-8.

Table 6-8. ITEP vs. non-ITEP Sized Vehicle Comparison

Performance Results	UH-60L Without ITEP	UH-60L With ITEP	% Change
VROC (ft/min)	726	1450	100%
Max Speed (kts)	147	157	7%

6.3.4 Robustness

For the above cases, the nominal values for the UH60 were altered by a percent change with the impact to forward flight speed and vertical rate of climb evaluated. By upgrading to an ITEP engine, the UH60 can sustain performance equivalent to the baseline configuration under harsher conditions. In the case of increased flat plate drag, for example, the forward flight speed of the UH60 with ITEP remains unchanged for less clean configurations (due to stall/compression dominating over parasitic drag) compared to the baseline, indicating that the UH60 with ITEP can endure aerodynamic deficiency in the way of externally mounted equipment, for example. Similarly, variation of air density and gross weight revealed significant performance for hover for the ITEP configuration, indicating robustness to atmospheric conditions and eight loading that cannot be accomplished by the baseline.
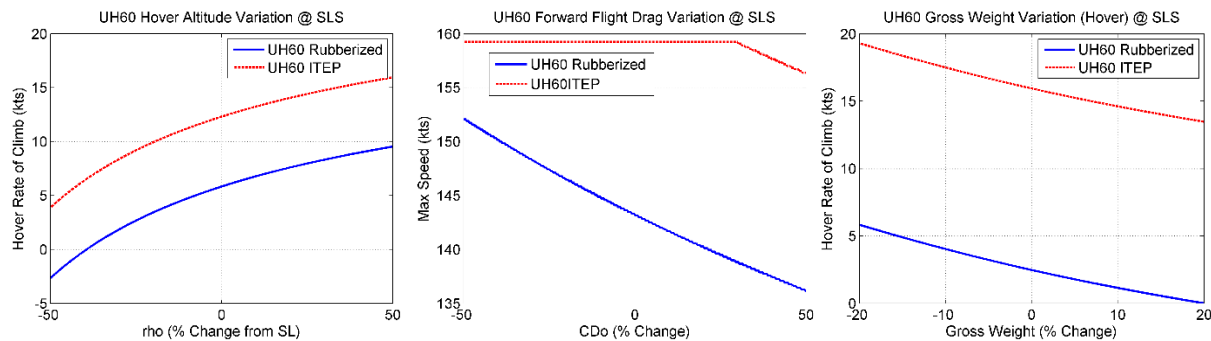
Figure 6-11. UH60 performance sensitivities to changes in drag, air density, and gross weight

6.4 Future Trade Studies

To further evaluate the engine upgrades, mission analyses will be conducted on top of individual sensitivity studies and power sweep curves. Given that operational range is a large factor of concern to the stakeholders, two missions will be analyzed for comparing baseline range: one in which both aircraft are loaded to MTOW (each carries as much fuel as possible), and one in which both aircraft carry the same fuel. This will allow for a comparison of operational performance and pure performance. Additionally, further studies into the impact of the higher power ITEP engine on the maneuverability and longevity of the UH60 will be analyzed using NDARC for cases of quick climb, dash, and turns.

6.5   Technology Evaluation

A suite of technologies has been researched for CATE. Table 6-9. includes the previous technologies of interest. These technologies and a growing list of developing technologies in Table 6-10 will be detailed and implemented into NDARC to examine the potential of applying it to the UH60. In addition, the technologies and new UH60 models can be implemented into an updated version of CATE.

Table 6-9. CATE Evaluated Technologies

Technology	Impact
CTEF (Continuous Trailing Edge Flap)	Reduce vibration, maintenance, and noise [24]
Plasma Flow Control	Increased payload capacity, higher speeds, and increased range [25]
Wide Chord Blades	Increased lift [26]
Leading Edge Slot	Delay retreating blade stall [27]
Individual Blade Control (IBC) i.e. RADICL	Suppress noise and vibration [27]
Swashplateless Rotor	Reduce complexity and improve reliability [27]
Advanced Actuator Technology i.e. Actuation Material in Airfoil Structure	SEE MORPHING BLADES
Hybrid Gears	20% reduction in gear weight [27]
Hover Infrared Suppression System (HIRSS)	Reduce IR signature [26]
Ceramic Matrix Composites (CMC)	Reduce fuel burn, emissions, and weight [28]
Helicopter Active Control Technology (HACT) i.e. Fly-by-Light/Wire Control System	Control system to improve all-weather/night mission performance [27]
Health and Usage Management Systems (HUMS)	Maintenance Reductions Unscheduled MMH/FH: -52%* Mission Aborts MMH/FH: -48%* Total MMH/FH: -17%* *Actual data from U.S. Army Deployed UH-60 Black Hawk helicopters with UTC Aerospace Systems HUMS [29]

Table 6-10. Extended List of Evaluated Technologies

Technology	Impact
Rotor chord extension	Tests with UH60A (Max benefits of 14% power reduction, or increase of 1300 lbs gross weight [30]
Rotor twist / camber morphing	Hover performance gains of 4 – 15% [30]
Variable span rotor	Tests with UH60A showed power reduction in certain conditions [30]
Reversible airfoils for stopped rotor	Used on NASA X-Wing [30]
Wing folding for compound helicopter	Early prototypes show reduction in downwash during takeoff [30]
Control reconfiguration	Reconfigure controls for situations of minimum power, minimum noise, or sudden failure. Tested with UH60. [30]
Advanced anti-torque	Noise reduction, safety, thrust vectoring for control. Seen on the Bell FCX-001. [31]
Composites	Weight reduction
Shrouded tail rotor	Increased anti-torque efficiency
Biplane stabilizers	Reduces aerodynamic penalties in low-speed flight and hover. Seen on Airbus H160 [32]

Information on morphing or reconfigurable rotorcraft was presented by Dr. Farhan Gandhi at a presentation at Georgia Tech title "Reconfigurable Vertical Lift". [30]

The technologies in question involve some with direct application to the UH60 upgrade and others with more potential for the FVL concepts. In either case, the research allows a more holistic comparison to be made inside NDARC by involving updated technology factors.

6.6 Configuration Comparison

Another significant concept is that the choice of the V/STOL configuration largely depends on the hover time required for the mission as seen in Figure 6-12. Design cruising speed can be related to the hovering time for each concept as seen in Figure 6-12. The shorter hover time configurations have lower cruise speeds, which is understandable as the requirement to hover usually requires more fuel, often leading to a larger engine selection to carry the weight, and then creating a geometry with a higher flat plate drag [33]. From the previous statements, it is understood that, just as the AHS competition is looking for, to design a system that is truly robust in a range of mission scenarios a reconfigurable system is required. The level of reconfiguration is of course at the hands of the designer and decision maker. Examples of reconfigurable rotorcraft using stowed-rotors include the Sikorsky stowed-rotor or the Lockheed folding blade concept [34]
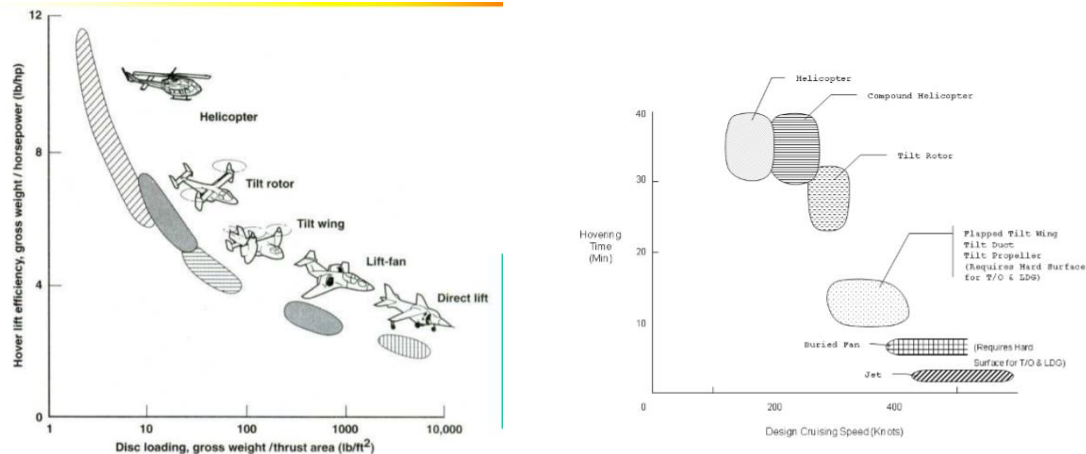
Figure 6-12: Operational ranges for traditional vehicle concepts

While investigating technologies and configurations three factors are key to finding a design which can meet hover and speed requirements of the future. The rotor tip speed must be reduced, the C_T/σ must be unloaded to some alternative form of lift, and the angle of attack of the rotor disc to the flight path must approach zero. [35]

In conclusion, the future of V/STOL aircraft requires advancements in five areas to reach proposed levels of operation. The difference in thrust required from vertical flight and cruise must be handled. During hover, a proper distribution of thrust must be situated for low-speed maneuverability. Internal complexity must be reduced as to lower empty weight. The addition of propulsors or more stringent operational areas must be met with higher fuel efficiency. Controls must be aimed at all regimes of flight for the safety of operating around urban areas and at lower altitude [33]. Future trade studies will look to take technology and configuration factors into account to compare the UH60 to FVL concepts. The goal will be to determine the areas in which an upgraded UH60 can achieve success with a lower cost and faster introduction timeline then FVL designs.

7. A Numerical Method to Calibrate and Forecast Technology Improvements for the UH-60 Helicopter Using NDARC

This work was presented at the 73rd American Helicopter Society forum in Fort Worth, TX. [36]

7.1 Introduction

A concept level tradeoff environment for future helicopters has been proposed in a previous paper. [1] The present document aims at performing the verification and demonstration of the environment by generating a use case for an existing aircraft: the UH-60. This paper proposes an alternative calibration process that aims at including data from additional sources, such as using optimization routines to minimize error between NDARC predictions and published data.

7.2 Calibration

Extensive details on NDARC calibration process and theory are found in NDARC theory and validation references [37] [38]. The present section will discuss a new way to construct the NDARC files to expand on what was proposed in a previous paper [39] by including new set of data.

Table 7-1 UH-60A/L/M Technologies Modeled

Aircraft	UH-60A	UH-60L	UH-60M
Engine	T700-GE-700	T700-GE-701C	T700-GE-701D
Improved Durability Gearbox		Installed	Installed
Wide Chord Blade system			Installed
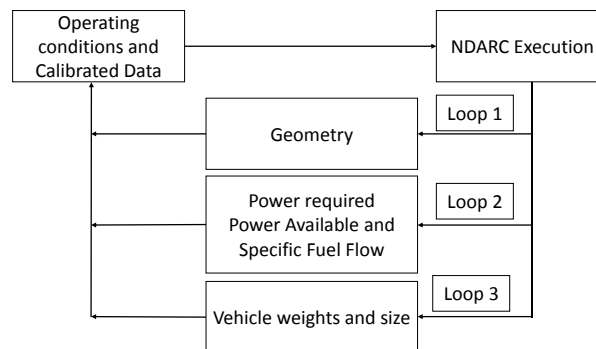
Figure 7-1 Calibration Process

Three calibration loops are performed consecutively as illustrated in Figure 7-1: the geometry is calibrated, then the power available and specific fuel flow are calibrated, and finally the vehicle weight and size are calibrated.

7.2.1 Geometry

Geometry for the UH-60A was derived using dimensions from the UH-60A mathematical model [40] and the UH-60A/L operator's manual [41]. This includes the aerodynamic coefficient and size of the fuselage and the tail, as well as the location of the main components.

7.2.2 Power Required

The calibration of power required involves many parameters. A typical model for power requirement decomposition in NDARC is detailed in NDARC's theory manual [38] and requires an adequate bookkeeping of power losses. The power required is a sum of installation losses, transmission losses, the accessory power and the power required by each rotor. In the case of a single main rotor configuration, two rotors are used: the main rotor and the tail rotor. The power required P_{req} by each rotor is a sum of induced power P_i , profile power P_0, interference power P_t and the parasitic power P_P which is the thrust multiplied by velocity, which includes the power to climb. The rotor can be modeled in NDARC by using a Table Performance Method or by using an Energy Performance Method. The Table Performance generates a linear interpolation for the induced power coefficient and the average drag coefficient of the rotor from a table. Both the induced power coefficient and the profile drag coefficient have to be generated from a higher fidelity aerodynamic code, such as CAMRAD or RCAS. The energy performance method generates the induced power coefficient and profile drag coefficient from a series of parameters. The energy performance method was chosen for this project due to the flexibility of the performance parameters that the method allows.

Two methods were selected to generate the rotor performance parameters: an automated calibration based on higher-fidelity aerodynamics results and a calibration of power required compared to the operator's manual operating points.

Energy Performance Method

The energy performance method is the process used in NDARC to evaluate the rotor power required. Within NDARC, the rotor power model is broken down into two independent design spaces: induced power and profile power. The induced power NDARC variables determine the calculation of the induced power coefficient (κ), while the profile power NDARC variables determine the calculation of the profile drag coefficient ($c_{d\ mean}$).

NDARC generates the coefficients based on a series of equations which can be found in the theory manual [38]. Among the factors of influence, there is variation with thrust coefficient, shaft angle, axial and

edgewise velocity and Mach number at the tip. Typically, users match NDARC predictions of the induced power coefficient and profile drag coefficients to the ones given by higher fidelity aerodynamic codes. The data required at various operating conditions is provided in Table 7-2. The information includes four independent variables (μ_x, μ_z, CT/σ, and MAT), and two dependent variables (values for profile drag and induced power coefficients from the comprehensive analysis tool).

Table 7-2 Information Required to Calibrate Rotor Model

Variable	Description
μ_x	Advance ratio along the x-axis
μ_z	Advance ratio along the z-axis
CT/ σ	Blade Loading (thrust coefficient / solidity)
MAT	Maximum Mach number at the advancing tip
Cd	Profile drag coefficient
κ	Induced power coefficient

The design space of this problem has the potential to become quite large, encompassing over 30 design variables for both induced and profile power. In addition, almost all of the NDARC variables must be treated as continuous variables over some practical range of values, further increasing the complexity of the problem. Consequently, the calibration of the parameters associated with the rotor energy performance method was posed as an optimization problem, with a genetic algorithm chosen to handle the various design variables.

Error calculations for both the induced power coefficient and profile drag coefficient serve as the objective functions to be minimized during the optimization. For both coefficients, the total error is calculated as the sum of the absolute relative error (summed over N calibration data points), where the value estimated from the NDARC curve fits is measured relative to the true value provided by either a comprehensive analysis tool or some other form of higher fidelity data. As the number of calibration data points, N, may vary from case to case, the objective functions in the optimization problem are represented as the average of this total error calculation, as shown below. This approach provides a metric to measure the calibration accuracy that is independent of the number of calibration data points used (i.e. the magnitude of the error does not scale directly with the number of calibration data points).

$$C_D Obj. Function = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \left| \frac{C_{D_{est}} - C_{D_{true}}}{C_{D_{true}}} \right|$$

$$\kappa \, Obj. Function = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \left| \frac{\kappa_{est} - \kappa_{true}}{\kappa_{true}} \right|$$

Operators Manual Power Required

A second technique to calibrate the rotor required power uses the operator's manual published data [41]. The proposed process is based on the work already published [39]. For the UH-60A, the data are in the form of engine torque as a function speed, aircraft weight, altitude and temperature.

The UH-60A operator's manual includes the torque per engine for vehicle gross weight from 13,000-22,500lbs, for various advance ratio including hover, from sea level to 20,000ft, at temperatures ranging from -50C to 60C. Calibration points were gathered by digitizing performance charts from the Operator's Manual for operations at Sea Level Static (SLS) and at 4,000ft, 95F condition.

A NDARC performance runs wrapper was created and the power required at each of the conditions was evaluated. The wrapper includes the generation of NDARC files from templates, the parsing of the output and the automatic execution of NDARC. A multi-objective genetic algorithm was used to minimize the two objectives given below by varying the parameters related to required power. A Non-Sorting Genetic Algorithm (NSGA) was chosen because it handles non-linear, discontinuous computation models and performs multi-objective optimization. For simplicity, the following two objectives are used:

1. Minimization of Root Mean Squared Error (RMSE) of power required to hover for gross weights between 12,000 lbs and 21,000 lbs, and at sea level standard (SLS) and 4,000 ft, 95F

2. Minimization of RMSE of power required in forward flight for set of forward speeds ranging from 0 to 155 kts at gross weights of 16,000 lbs and 18,000 lbs, and at SLS and 4,000 ft, 95F

The optimizer generates a Pareto frontier of possible combination of parameters that represent tradeoff between modeling error in hover and in forward flight. One cannot choose to reduce the modeling error in hover without worsening the modeling error in forward flight. A multi-objective decision-making technique, TOPSIS, was used to select the combination of parameters that represent a good tradeoff between both cases.

7.2.3 Engine

NDARC allows for two types of engine models suitable for the UH60 model: the turboshaft engine tabular model and the Referred Parameter Turboshaft Engine Model (RPTEM).

The turboshaft engine tabular model is comprised of tables of Power available and fuel flow, as a function of altitude, flight speed and rating. In the case of this research, the required data could have been extracted from the operator manual, for example.

The RPTEM consists of a set of physics-based equations that provide the power available and the performance at power required based on calibration factors. The RPTEM was selected for this research as it provides a flexible way to represent the engine and apply some technologies, which can be scaled and modified throughout the process. Around 20 variables can be modified to calibrate the engine models. The calibration was made in two steps: first, the power available is calibrated, and second, the fuel flow is calibrated.

7.2.4   Engine Power Available

In a first step, the power available parameters were calibrated for various flight conditions documented in the operator's manual, including the vehicle weight. The process is based on published work [39]. Power required points were digitized from the operator's manual, and a NDARC performance evaluation was performed at each point. A genetic algorithm was used to minimize the error function in power available, by changing the various parameters affecting power available of the main rotor. At this point, efforts to adequately bookkeep between of the power available and the power losses is important.

7.2.5   Engine Fuel Flow

In a second step, the engine fuel flow was calibrated against the same documented flight conditions. Because the fuel flow is dependent on the required power at that flight condition, the error on power required predictions would affect the fuel flow, which would additionally affect the fuel flow calibration. Consequently, a true function of the fuel flow against power required was created. Once again, a genetic algorithm as was used to minimize the error between the true function and the generated function by NDARC.

7.2.6   Weight and Sizing

Once the Engine and rotor performance models are calibrated, the helicopter is sized to perform the expected mission. The sizing task in NDARC internally converges on the vehicle size and weight in order to successfully perform the mission. In order to obtain the correct vehicle size, weight factors are modified.

To complement the actual calibration process illustrated in the NDARC theory [38], an optimizer is used to minimize the discrepancy between the vehicle weight and the NDARC predictions. The aircraft weights were based on published UH-60A weight breakdowns of an actual production helicopter.

7.3 Calibration Results

7.3.1 Engine Calibration

This section details the results of the T700-GE-700 engine calibration from the method discussed in the previous section. The calibration of the T700-GE-701C is also performed independently and as a derived configuration from the T700-GE-700 in the next section *Technology Infusion.*

First, the T700-GE-700 engine was calibrated as per described in the previous section. The results of the power available as a function of various altitudes, temperature and airspeed match well with the published data. The fit is characterized namely by a RMSE of 35 hp (total) for the power available for the various operating conditions, and a RMSE of 0.048 lb/(hr hp) for the specific fuel flow.

Similarly, the GE T700-GE-701C was calibrated using the operator's manual published data. Similar to the previous engine model, an RMSE of 37hp (total) was found between the NDARC optimized calibrated model and the operator's manual data, and 0.083 lb/(hr hp) for the specific fuel flow.

7.3.2 Power Required Calibration

The two power required calibration methods proposed in the previous sections are conceptually very different from one another. In the "Energy Performance Calibration Process", the data comes from higher fidelity aerodynamic codes which separated the induced and profile power of the rotor. In the "Operator's Manual Power Required" calibration process, the data comes from the overall torque required as a function of the flight condition, which includes all sources of power required for the main and tail rotor.

In order to compare the models, a vehicle performance evaluation was performed with aircraft models generated by the two methods. The "Operator's Manual Power Required" calibration method led to good agreement between the NDARC results and the operator's manual. The RMSE of the power in hover is 6.3hp and 41.5hp in forward flight.

Due to limited access of high fidelity data, the aerodynamic data of a UH-60 in hover was extracted from a published paper [37] [42]. The Energy Performance Method model calibration result is shown in Figure 7-2 and in Figure 7-3. The root mean squared error (RMSE) of the induced power coefficient is 0.004 and 0.02 for the profile drag. The results show that the optimizer is successful at matching the provided data set.
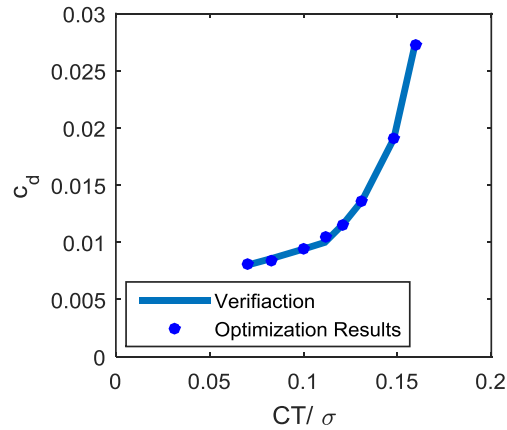
Figure 7-2 Optimization results: Profile drag coefficient in hover and the verification data from [37]
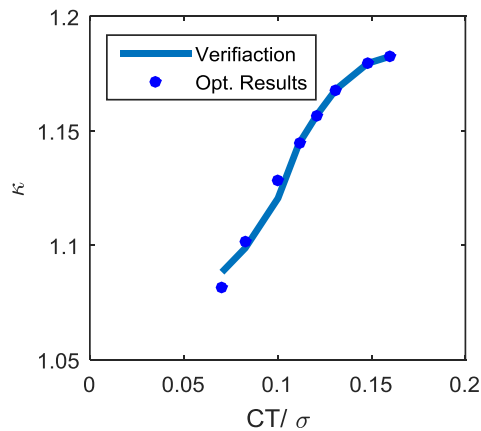
Figure 7-3 Optimization results: Induced power coefficient in hover and the verification data from [37]

The aircraft description file was modified to include the parameters found with this technique. This method leads to an under prediction of the power to hover at SLS and 4,000ft, 95F, with a RMSE of 32hp (total) for the various GW expressed in the operator's handbook. The NDARC performance runs at various CT/σ, in SLS conditions show that the induced power coefficients are identical between the two techniques. However, Figure 7-4 shows the discrepancy between the profile power coefficient. The discrepancy is relatively small, and is constant for the three cases illustrated in the figure. This difference could possibly be a result of how the power was bookkept in the Operator's Manual calibration technique, and could be reduced if more information was available on the other loss mechanisms, such as transmission losses, etc.
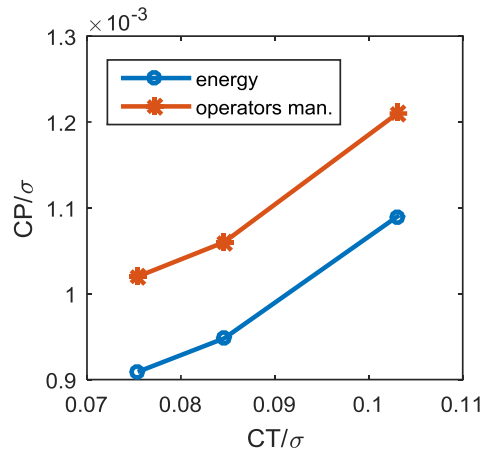
Figure 7-4 Profile power coefficient comparison between the Operator's Manual Power Required Motor Model and the Energy Performance Method Model

The results show a relatively interesting potential for this technique and relatively good agreement between the two calibration techniques. It also opens to more test cases, including some operation in climb and in forward flight.

Due to the lack of available data for advancing flight, the results from the "Operator's Manual Power Required" calibration process will be used as the principal helicopter model for the following sections of this paper.

7.4   Technology Infusion

The technologies implemented on the UH-60 variants are illustrated in Table 7-1. In an attempt to mimic the prediction of the UH-60L and UH-60M performance and sizing, the model of the UH-60A is modified with the technology impact factors to represent the respective technologies. The following section details how the technology impact factors were calculated and how their impacts are propagated in a sizing and performance evaluation environment.

For the engines, documents [43] gave values for the weight, engine ratings and fuel consumption. NDARC was coupled with an optimization routine that acted as a numerical solver to calculate the values for the technology impact factors. It was assumed that each engine had some level of technology that impacted engine weight. After inputting the intermediate rating power (IRP) for the engines into NDARC, an optimization routine was used in which the engine weight technology factor was varied until the output engine weight matched the data. For the T700-GE-701C, the calibration factor used for estimating the original T700-GE-700 engine weight resulted in an output engine weight of 457.4 lb, which is close to the 458 lb value given by the technical specifications. For the T700-701D, the optimization routine indicated that the technology of the T700-701D engine resulted in a 3.6% weight decrease. Neither reference

122

indicated changes to the entire propulsion group (changes in nacelle and structural weight), so NDARC parameters relating to these groups were left unchanged.

The information found regarding the improved durability gearbox (IDGB) indicated the increased power rating, but no information about efficiency or weight was found. For performance analysis in NDARC, the IDGB is simply represented by increasing the transmission ratings. For sizing analysis, performance requirements such as MTOW or VROC are used to calculate the drive system rating, so there are no technology impacts to evaluate for sizing purposes.

A different approach was taken for evaluating the wide chord blade (WCB) system to demonstrate how to make predictions when technology is still being developed. The WCB is a new rotor blade, with new planform, a wider chord and different airfoils. Yeo et al. modeled the WCB system in a high-fidelity code and found that the increase in solidity was a main performance driver as a result of the de-loading the blades. [42] Due to the absence of the results of high fidelity aerodynamic simulation, only the change of solidity is implemented. The 10% increase in solidity reported for the WCB by Yeo et al. results in a 9.1% decrease in blade loading.

Blade weight changes were based on results from a NASA project, which focused on modeling the structure of a composite rotor blade and using optimization to find minimum weight designs. [44] This research used the UH60A as a validation case. Nixon's results for estimating blade weight changes due to composite designs were based on the aerodynamics of the UH-60A. Nixon's paper concluded that a single-spar composite design would result in a 21.3% weight reduction and a multi-spar composite design would result in a 12.1% weight reduction relative to the metallic design used for the UH-60A. There was no specific information found on how the control weight would change, so no assumptions were made as to potential technology impacts for these. The technology impact factors are summarized in Table 7-3, and includes the more conservative blade weight impact factor.

Table 7-3 UH-60A/L/M Technology Impact Factors

Technology	Blade Loading	Engine Weight Factor	Blade Weight Factor
T700-GE-701C	0%	0%	0%
T700-GE-701D	0%	-3.6%	0%
Improved Durability Gearbox	N/A	N/A	N/a
Wide Chord Blade System	-9.1%	0%	12.1%

The operator's manual gives information regarding UH-60L fuel flow, allowing to verify the engine fuel consumptions made earlier. The operator manual's data on power required was used to compare the

NDARC model predictions. Finally, high-level mission performance characteristics of the UH-60A, UH-60L, and UH-60M models were used to verify the models. This data was used to indicate how close the NDARC performance models represented the various Black Hawk models.

Though there are no technology factors to attribute to the UH-60L model, it is still modeled in NDARC by increasing the engine power available. Doing so also increases the engine weight and represents the T700-GE-701C, and increasing the drive system limit represents the IDGB. The power required output was compared with data from the operator's manual.

In general, power required was overestimated, which will underestimate the performance calculations for maximum speed, ceilings, hover VROC, or maximum gross weight. NDARC estimates component reference surface area based on weight, so the heavier engine results in more drag, which will cause increased power required. Nothing was known about the change of drag coefficient of the UH-60L, so no change was made on this aspect. Fuel flow was underestimated, which will cause mission range or endurance to be potentially overestimated. No assumptions were made about fuel flow of the new engine, so the consistent underestimation indicates that the new engine burns more fuel. It is expected that the RSME will increase as other vehicles are modeled.

Table 7-4 NDARC Weight Predictions Error for the UH-60L

Component	Error between NDARC and production UH-60L
Rotor Group	3%
Empennage Group	3%
Fuselage Group	25%
Structure Total	14%
Engine System	11%
Propulsion Total	3%
Empty Weight	9%

Modeled weight predictions for selected groups were compared with reported weight values for the UH-60L and the error are illustrated in Table 7-4. Note that no tech factors were changed between the UH-60L and the UH-60A NDARC representation models, and that only two technologies were implemented on the UH-60A to represent the UH-60L. Upgrades to avionics, crashworthiness, hover infrared suppression systems, and any other modifications in the block upgrade were not included. The technologies were not included to reduce the scope of the problem to only performance related technology. Thus, it is expected that there will be error in the weight estimates.

A large source of error in the fuselage weight drove the high error in structure group weight. Similarly, the engine system weight error drove the propulsion system error. The UH-60L incorporated many things from

the variants developed from UH-60A (the Navy and Air Force utility helicopters) in addition to general weight creep, so it is possible that investigating what these changes were and how they affect fuselage weight would allow for better predictions. The error of the engine weight was negligible (less than 1%). The error of the engine system group weight is largely due to underestimating the exhaust system weight by 66.7%. It is likely that an updated hover infrared signature suppressor (IRS) increased the weight of the exhaust system. Information relating to hover IRS is generally restricted so it was not selected as a technology to investigate. More information about technology upgrades will result in better predictions, but the results show that the system sizing can still be represented with a limited information. The error in the results above stems from a partial representation of the block upgrade. However, the model still allows for inferences about the two upgrades that were applied. This model can be used to answer questions about how a new transmission and engine will affect the useful load of the black hawk helicopter.

To represent the UH-60M, the UH-60L file was modified. Similarly for the UH-60L, the engine available power was increased and the -3.6% technology factor was applied to the engine technology factor to represent the T700-701C. Additionally, the rotor blade loading was decreased by 9.1% and the main rotor weight technology factor was decreased by 12.1%. No other changes were made to represent the UH-60M.

Table 7-5 NDARC Performance Analysis of UH-60A, UH-60L, and UH-60M vehicles

Model	VROC (ft/min)		Vcr (kts)		MTOW (lbs)	
	Data	NDARC	Data	NDARC	Data	NDARC
UH-60A	377	0	140	142.5	20,250	20,620
UH-60L	1315	412	155	151	22,000	23,505
UH-60M	1646	862	151	153	22,00	23,406

Table 7-5 gives the results of the NDARC performance prediction of the UH-60A, UH-60L, and UH-60M along with the published results. [45] This reference was used because data on the UH-60M similar to the data used for the UH-60L and UH-60A is not available to use for comparisons. The vertical rate of climb (VROC) and maximum speed performance estimates are for a gross weight of 16,800 lbs at 4000 ft, 95F. For VROC, maximum available power is 95% IRP and for maximum speed, maximum available power is 100% MCP. The reference lacks the important operating condition information about the conditions of the performance points. However, it was used across all vehicles so that similar assumptions were used.

There were significant errors in estimating the VROC. In the UH-60A case, there was not enough power available to hover at 4,000 ft, 95F, returning 0 for VROC and indicated that power available was exceeded. Further investigation of this operating condition results needs to be performed. NDARC has acceptable estimate of maximum speed and overestimates the MTOW for all cases. For the maximum speed, NDARC

is correctly giving power available due to ram effects increasing power. For the overestimates of MTOW, there was a lack of information relating to the environmental condition in the documentation. In the operator's manual, the closest condition for reaching the UH-60A max weight was SLS and at 100% IRP or the drive limit. Without knowing the MTOW conditions for the reference data point, it is difficult to make conclusions regarding the accuracy of NDARC's predictions. [45] Additionally, it is unknown if MTOW is limited by structural safety margins. However, the NDARC predictions do agree with the trends between the models. The VROC increases for each new model while the maximum speed and MTOW only see real increases between the UH-60A and the UH-60L models. Technology prediction focuses on the changes between models since models are inherently wrong. The previous table indicates that the technology analysis and modeling environment are capturing these changes, meaning that this NDARC method is acceptable to use in a vehicle development scenario.

7.5   Conclusions

This paper presented new approaches to calibrate rotorcraft performance models in NDARC. In most techniques, an optimizer was used to obtain the model parameters. The use of the optimizer reduced the input of the user during the process. However, it was noted that setting up the optimization problem requires experience from the user to choose the model types, which variables to change and to bound the problem by providing adequate limit and initial guesses on variables.

The engine deck was built by reducing the discrepancy between operator's manual power available and fuel flow and the NDARC model, using an optimizer that varied the RPTEM parameters. Good fit was obtained for power available and fuel flow for the T700-GE-700 and the T700-GE-701C.

The power required calibration was made by two methods. First, the rotor parameters were modified to minimize the error between published power required from the operator's manual. Then, minimization of the error between modeled and published profile drag coefficient and induced power coefficient was performed in hover. Both methods led to similar results for hover, which opens to more verification cases.

8. Development of a Framework for Mission and Operational Modeling

This work was presented at the 73rd American Helicopter Society forum in Fort Worth, TX. [46]

8.1   Introduction

The loss of hundreds of rotorcraft aircrews during the conflicts in Afghanistan and Iraq has motivated the Future Vertical Lift (FVL) initiative, a plan to develop the next generation of military rotorcraft. Key goals of the initiative include developing "the most capable aircraft at the best value by minimizing development, acquisition, and life cycle costs through Joint solutions of common core technologies, architectures, and training, emphasizing the ability to conduct safe, reliable and continuous operations". [47] The goals of the FVL initiative highlight both tradeoffs and opportunities for technology infusion, at both the component and system architecture level that have a strong impact on system capability, reliability, and life cycle cost.

Estimates of traditional rotorcraft performance metrics are often available during the conceptual design phase and are obtained using rotorcraft performance analysis and sizing tools like the NASA Design and Analysis of Rotorcraft (NDARC) environment. [38] The NDARC environment was built with the intent to perform trades for technology infusion with respect to vehicle performance metrics including weight, size, range, and endurance. However, NDARC is unable to assess the impact of technologies on key system sustainment metrics including reliability, availability, maintainability, and affordability, which are key to FVL initiative goals. Reliability is the probability that the aircraft system will operate without failure during a given time period at specified conditions while Operational Availability represents the percentage of time that the aircraft system is operationally capable of performing a mission assigned to it. [48] Armstrong et.al. note that maintainability is a measure of the cost, time, and effort required to maintain the desired level of system reliability and availability. [49]

System reliability, availability, and maintainability are related to the Operations and Support (O&S) cost, which contributes to approximately 70% of the system lifecycle cost. [50] The DoD Reliability, Availability, Maintainability (RAM), and Cost Rationale Report Manual highlights the importance of incorporating sustainment metrics early in system design because it "enables the acquisition and requirements communities to provide a weapon system with optimal availability and reliability to the warfighter at value." [48]

This work is a continuation of the work of Velden et al., presented previously at the Rotorcraft Virtual Engineering Conference. [51] The work of Velden et al. described an overall integrated simulation environment that encapsulated system capability, availability, and affordability, and presented results for availability and Mean Time Between Mission Affecting Failures (MTBMAF) for the UH-60M when performing a standard utility mission. However, this simulation effort did not include the effect of

maintenance actions or downtime on the rotorcraft availability. Previous work by Armstrong et al., presented at the AHS 72nd annual forum included a maintenance manager to study the downtime incurred due to part failure and replacement. [49] The work of Armstrong et al. modeled vehicles as a container of parts that accrued damage during normal operations. The method implemented also allowed for the incorporation of technology factors to explore the effects of technology infusion on operational availability, vehicle loss rate, and the operations and support costs. Results representing the availability and O&S costs for the UH-60M were presented.

This paper describes the development of an integrated simulation environment in the form of a discrete event simulation that tracks the long-term, steady-state Operational Availability and Maintenance Free Operating Period (MFOP) of a rotorcraft system. A maintenance manager is included to aid in the study of system downtime due to part failure and replacement while the rotorcraft system architecture is represented by a series of event trees unique to each portion of a phased-mission profile. Developing a modular simulation framework for investigating the Operational Availability and MFOP of a new rotorcraft system acts as an enabler for component technology and system architecture trade studies; this can be expected to be beneficial to achieving the goals of the FVL initiative.

8.2 Simulation Environment Development Methodology

8.2.1 Conceptual Approach

The model is intended for use in simulating the usage of a single aircraft. In this model, the vehicle performs a mission, is evaluated in a post-flight inspection, and either goes to maintenance or begins the next mission based upon the presence or absence of failed systems as shown in Figure 8-1.
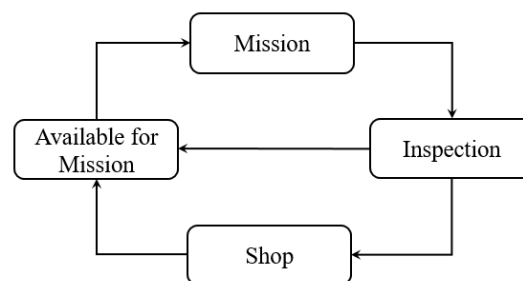
Figure 8-1. Conceptual Model Description

Both mission phases and maintenance actions occur at discrete time intervals throughout the simulation. The vehicle may be sent to maintenance for two reasons, a condition-based repair or replace action noticed during routine inspection or an in-flight failure identified during post-flight inspection or during flight that resulted in a mission abort. [52] In both cases, maintenance is a result of a trigger that occurs at a discrete time during a phased-mission. Because part hours are tracked at discrete time steps, constructing an

integrated discrete-event simulation is appropriate. Further, uncertainty in both individual part life and the time required to repair or replace components results in the need to introduce part life and repair time distributions rather than setting deterministic values for individual components. To capture the effect of the distributions on key parameters used in the models, Monte Carlo studies are performed to capture the variability of the key metrics tracked in the simulation environment.

8.2.2   Metrics of Interest

The key metrics tracked in the discrete-event simulation are related to reliability, availability, and maintainability of the system. These metrics are selected because they are directly related to the operations and support costs for an aircraft. The key metrics tracked are shown in Table 1.

Top-down assessment of reliability, availability, and maintainability metrics relies on subject-matter experts and is qualitative in nature, a process that requires significant documentation to remain transparent. Further, performing tradeoffs from a top-down perspective with expert-in-the-loop evaluation methods is not practical. Evaluating reliability, availability, and maintainability metrics from the component level, a bottom-up approach, based on data obtained for individual components allows for an easily documented process that can be performed numerous times by an analyst, and results in a preliminary quantitative prediction as opposed to a qualitative comparison between architectures.

Table 8-1. Metrics Tracked in the Simulation

Parameter	Units	Description
Operational Availability (A_0)	%	The percentage of time that the vehicle is operational – capable of flying missions
Maintenance Free Operating Period (MFOP)	hr	The number of hours a system can complete its assigned missions without required maintenance or restrictions due to system faults or limitations [53]
Cost to Replace Failed Parts	$	The total cost of replacing failed components
Cost to Replace Parts on Condition	$	The total cost of replacing parts due to deterioration of the part
Maintenance Man Hours to Replace Failed Parts	hr	The maintenance man hours required to replace failed components
Maintenance Man Hours to Replace Parts on Condition	hr	The maintenance man hours required to replace parts due to deterioration of the part
Number of Failed Parts	Parts	The number of failed components of each type replaced during a simulation
Number of Condition-Based Replacements	Parts	The number of parts replaced due to part condition

8.2.3   Modeling Approach

The discrete-event simulation is developed for a single aircraft. A phased-mission with phase-dependent event trees for mission-critical and safety-critical failures is defined using an interface and loaded into the model. Component MTBF values with associated distributions are also loaded into the model. The single vehicle flies missions according to Figure 8-1 and interfaces with the maintenance manager that determines the amount of downtime incurred due to maintenance requirements. Each of these components ultimately feed into the evaluation portion of the simulation that determines the MFOP, mean time between system failures (MTBF), and the operational availability (A_o). The framework of the simulation is illustrated in Figure 8-2.

The key simulation parameters are the number of hours for which the simulation will be run, or alternatively the number of MFOP cycles that will be simulated. Additionally, to account for the distributions included for component MTBF and MTTR values, the number of Monte Carlo cycles performed is also specified.
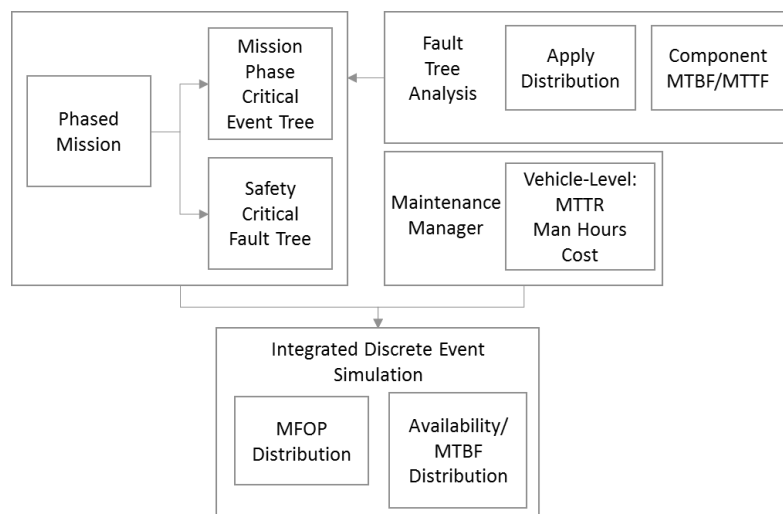
Figure 8-2. Simulation Layout

8.2.4   Phased-Mission Modeling

The mission used for evaluation of the system consists of a single phased-mission. The phased-mission can be adjusted to model civilian transport missions or military missions including scout, attack, and transport or medevac, allowing for significant modeling flexibility. The key parameter defining a mission phase is its elapsed time. Additional parameters regarding component age and failure propagation are also required for each mission phase which are discussed in detail in the subsystem modeling approach section.

Implementing a phased-mission has several advantages. One of these advantage is related to the individual aging of the constituent aircraft systems. Using a phased-mission, active components are aged during a given mission segment while inactive components do not receive hours, the metric used to track component age. This allows for a more-accurate model of the effect of mission-specific component use than simply tracking cumulative age. Further, implementing a phased-mission allows for the definition of how component failures propagate during each mission phase due to the criticality of that component to safety or mission success. [53] During the mission, the consequence of a component failure on mission success and vehicle survivability depends on the mission phase.

Although there are other advantages, using a phased-mission is required for proper evaluation of the MFOP for a particular aircraft system. The concept of MFOP is defined by Mitchell as a period of operation where a system must be able to complete all of its assigned missions without required maintenance action or restrictions on the operator due to system faults or limitations. [54] Here a phased-mission is critical because the MFOP concept is dependent upon the completion of a specific mission profile. The criticality of a failure with regard to the mission or vehicle safety is dependent upon the defined mission.

8.2.5   Modeling the Vehicle Systems

Modeling the vehicle accurately, and with a variable level of fidelity, is critical for the design trade studies envisioned at the subcomponent and architectural level. For the subsystem trades, it is important to capture information regarding each of the components in the subsystem as well as the interconnections between those parts. Capturing the component interaction at this level is required to allow for component trades or technology infusion at the subsystem level. At the architecture level, the interconnections between relevant subsystems drive the trades.

The work of Armstrong models the vehicle as a set of parts, each with individual properties that specify how the part fails, when it will fail, and the cost and time required for repair or replace actions. [49] Here, a similar approach is taken in decomposing the system into constituent components with specific properties. The parameters required to define each component are shown in Table 8-2. Unlike the approach of Armstrong, each of the parts belong to event trees that describe the effect of failure on the system, specifically whether it is a safety-critical failure, a mission-critical failure, or does not affect the mission but must be repaired after returning from the mission.

Table 8-2. System Definition Inputs

Parameter	Units	Description
MTBF	hr	The mean time between failures for a repairable component, alternatively MTTF for a non-repairable component
MTTR	hr	The mean time to repair a component, including removal, repair or replacement, and installation to the vehicle
Start Age	hr	Starting age for a component that is installed on the vehicle
Time to Repair Model	N/A	Model (Gaussian, Weibull, Exponential, or Lognormal) and necessary parameters to assign repair times for components
Component Failure Model	N/A	Model (Gaussian, Weibull, Exponential, or Lognormal) and necessary parameters to assign failure times for components
System Cost	$	The cost to repair a system
Component Life	hr	The manufacturer-determined life where a component will deteriorate such that an inspection of it will require repair or replacement
Inspection Time Windows	hr	The window used in the field or shop to determine whether an inspection should be undertaken to evaluate part condition

A notional example of an event tree for a mission phase is shown in Figure 8-3. This example shows only a few components, but is intended to highlight the key features of the modeling approach. 'And' gates are implemented to require the failure of two or more items concurrently for the failure to propagate to the top level and signal a mission failure in a specific mission phase. 'Or' gates allow for the propagation of a single failure upward in the event tree.

The organization of components using event trees to capture failures during a given mission phase allows for the capture of failures that may not be mission or safety critical when considered in isolation, but contribute to the workload experienced by maintenance personnel.
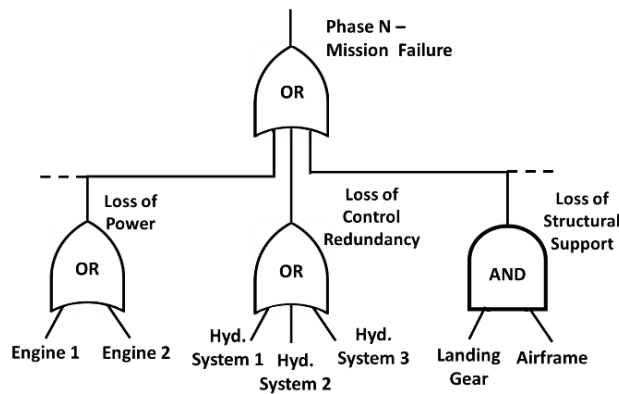
Figure 8-3. Notional Event Tree

8.2.6 Maintenance Manager

A maintenance manager is implemented to determine when the helicopter requires maintenance as well as how that maintenance is performed. Service may be required in two distinct cases. The first case is when a part deteriorates and must be replaced when found during a routine inspection. In this instance, no failure is experienced by the aircrew during a mission. The second case occurs as a result of an in-flight failure triggered by a component exceeding its operable life assigned based upon the MTBF distribution specified for the part in the simulation inputs.

If the maintenance manager determines that the helicopter requires maintenance after a mission, the first action is to determine if the helicopter needs to undergo additional preventive maintenance. By checking to determine if parts with specified inspection windows must be replaced soon, the amount of time that the helicopter must be removed from service can potentially be reduced.

After determining the key components that must be maintained or replaced, the maintenance tasks are assigned to available maintenance personnel on a longest-first basis. Using this approach, the total downtime is minimized as the amount of work performed concurrently by the mechanics is maximized. Ultimately, the maintenance manager is key to determining the downtime, maintenance man hours, and cost incurred due to necessary maintenance actions. The downtime calculated by the maintenance manager is based on delay time to acquire the part and necessary tools, the component MTTR, personnel availability, shop availability, and the number of maintenance personnel in the shop. A flowchart describing the inputs and outputs of the maintenance manager is shown in Figure 8-4.
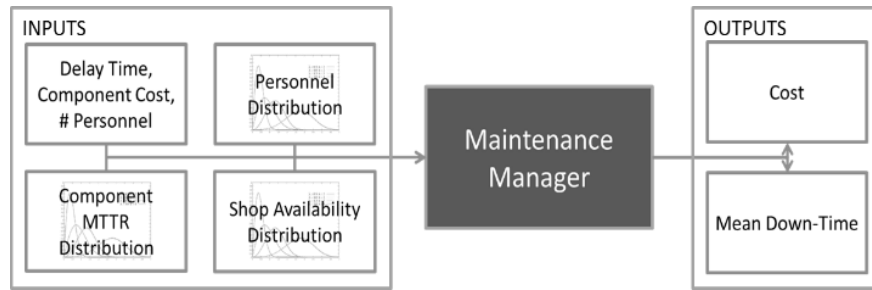
Figure 8-4. Structure of the Maintenance Manager

The key inputs to the maintenance manager are briefly discussed in Table 8-3.

Table 8-3. Maintenance Manager Inputs.

Parameter	Description
Shop Availability Distribution	Defines parameters for a distribution that describe whether the shop is available
Personnel Availability Distribution	Define parameters for a distribution that describe whether a maintenance person is available
Number of Maintenance Personnel	Total number of maintenance personnel assigned to the vehicle
LDT Factor	Multiplicative factor on downtime used to model LDT
Required Maintenance Actions	Maintenance actions that are required when the vehicle enters the shop
Preventive Maintenance Parameters	The time when the part will be revealed to need repair or replacement during specified part inspection intervals
Preventive Maintenance Switch	Switch that may be toggled to turn off all forms of preventive maintenance

The maintenance delay time incurred due to the acquisition of a part or specialized tools as well as the setup time required to perform a specific maintenance action is known as the Logistics Delay Time (LDT) and is largely based upon the logistical support network and environment of operations. [55] Estimating the LDT is beyond the scope of this model as an accurate estimation of this value would require in-depth modeling of the logistical pathways used for part transfer as well as the initial distribution of parts across multiple maintenance facilities or warehouses. LDT is exceptionally difficult to estimate and will change significantly over the lifetime of a vehicle. [56] For example, during introduction, the supply system will not be fully-stocked with spare parts for a given aircraft. Further, under combat conditions, estimating the LDT is not feasible in most cases. For a well-stocked maintenance facility, away from combat operations or in a civilian context, the LDT will be small. For this model, it is assumed that this is the case. LDT is assumed to increase the total downtime by 10%. However, it is acknowledged, that in cases where a maintenance facility is not well-stocked, LDT can quickly balloon to become the largest component of downtime. [56] For the discrete event simulation constructed, LDT is included in the model by including a

134

multiplicative factor on the downtime reported by the maintenance manager, in this case the factor is 1.1, representing a 10% increase in downtime due to LDT for a well-stocked maintenance facility.

Another form of delay incurred is known as the Administrative Delay Time (ADT). ADT is a component of downtime not spent waiting on spare parts, instead the delay is due to a lack of maintenance personnel or other non-part-related resources. [57] Because the maintenance manager considers shop availability and the availability of maintenance personnel, ADT is modeled directly through consideration of shop personnel and availability rather than a correction factor.

During calculation of the downtime incurred due to maintenance actions, two metrics that are key drivers for the maintenance cost are tracked: total part cost, and the maintenance man hours. Because preventive maintenance is considered in this simulation, the cumulative part cost and maintenance man hours are tracked for both preventive maintenance actions and maintenance due to during-mission failures. Tracking cost and maintenance man hours for preventive and failure-based maintenance separately allows for trades to be performed on the parameters that govern preventive maintenance where otherwise the necessary data would be inseparable from a single cost metric.

8.2.7   Modeling Technology Impacts

Using the developed discrete-event simulation to evaluate a given system architecture with a known set of subsystems is valuable and does provide insight into a system. However, the discrete-event simulation is also valuable in quantitatively modeling the effects of technology infusion on an in-service vehicle system.

For technology infusion at the component level, a model may be built for the baseline aircraft and modifications made to the system of interest to include the effects of technology infusion on that system. These modifications may come in the form of adjusting the distributions for the MTBF, the MTTR, or both, in addition to the cost of the system. This allows for a one-to-one comparison on the effect of technology infusion at the component level on vehicle operational and supportability metrics. It is difficult to accurately identify the MTBF, MTTR, and cost values for a given set of subsystems if data is unavailable. [56] However, by constructing a baseline and a technology infused variant using the best available approximations, the effect of the subsystem data takes a secondary role to the technology impact. This occurs because the subsystem data is consistent between the models and key differences will arise due to the infusion of a component-level technology.

Evaluating the effect of technology infusion at the component level on operations and supportability metrics for in-service vehicles is critical because it allows decision makers to see the true impact of upgrades. Although upgrades may provide performance benefits like reduced fuel burn, considering how these

135

systems affect supportability metrics must also be considered to determine how the upgrade impacts the vehicle system.

8.2.8   Modeling Architectural Tradeoffs

The developed discrete-event simulation is also useful in performing architectural tradeoffs during system design. Early in the system design process, tradeoffs of system architecture with regard to reliability, availability, and maintainability are critical to selecting both components and the architecture that meet performance and sustainability metrics.

Utilizing the discrete-event simulation to evaluate candidate architectures at the conceptual design level allows for consideration of novel concepts as well as more specific architectural decisions that may include component redundancy for example. The key to utilizing the discrete-event simulation to evaluate candidate architectures with regard to operations and supportability metrics early in design is the flexibility to define systems and architectures that are representative of many points throughout the design space.

8.3   Model Implementation

The discrete-event simulation is applied to a Bell 206 civilian helicopter. Dougherty notes that the Bell 206 is one of the most commonly used aircraft by flight hours. [58] In addition to being pervasive in the civilian market, the Bell 206 is closely related to the OH-58, a military rotorcraft typically used for scout missions. Due to the limited availability of specific data for the Bell 206, the implementation documented in this publication may be best thought of as modeling a notional generic helicopter similar in features to a Bell 206.

Implementing the developed model for the notional aircraft is a multistep process which requires definition of the system architecture, the mission, and component attributes including failure densities, repair times, and cost data.

8.3.1   Model Implementation

The purpose of developing an architecture for the aircraft of interest is to understand, at an appropriate level, the systems that are present in the aircraft as well as the linkages between the different components so that the event trees similar to those shown in Fig. 3 may be defined.

The methodology used to develop the event trees is based on the process used to complete a functional hazard assessment, "a systematic, comprehensive, examination of functions to identify and classify failure conditions to those functions according to their severity." [59] The process of completing the functional hazard assessment is as follows:

1. Identify all functions associated with the system under consideration
2. Identify and describe the failure conditions associated with each function
3. Determine the effect of the failure conditions
4. Classify the failure condition by its effect on the aircraft

Based on the process outlined to perform a functional hazard assessment, the process to define event trees starts with performing a functional decomposition of the notional rotorcraft. Using the developed functional architecture, individual components are enumerated and assigned to specific functions. Using interconnections between components defined in the functional and physical architectures, subsystem or component failures are defined to be safety or mission critical. The process is illustrated in Figure 8-5.
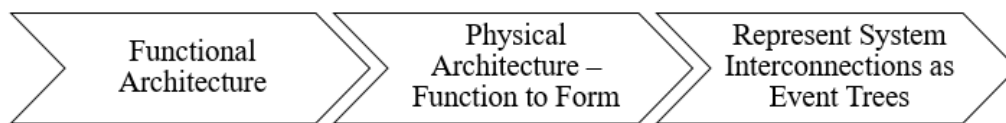
Figure 8-5. Process Used to Define Event Trees

A functional architecture for a notional single main rotor helicopter in the cruise mission segment is developed and is shown in Figure 8-6. **[60] [61]** Figure 8-6 is not meant to be exhaustive, rather it is intended to capture the most critical functions. In some cases, multiple functions map to a single component and in other cases, a single function maps to multiple components. The key components during the cruise mission phase are included in a notional physical architecture provided in Figure 8-7. **[60] [61] [31]** Although not shown in this paper, functional and physical architectures for the notional helicopter are also developed for additional mission phases including startup, takeoff, setdown, and shutdown.

The physical architecture of components required during the cruise mission phase can be broken down into two distinct event trees, also commonly called fault trees. The mission abort event tree defines the failures required to trigger a mission abort, while the safety critical event tree defines the failures that result in a vehicle crash. An example of a mission abort tree and a safety critical event tree for the cruise segment is shown in Figure 8-8. These event trees are also developed for the other mission phases including startup, takeoff, setdown, and shutdown.

It is worth noting that based on component type, specific failure modes will exist. The failure modes of a single component may be very different in nature as shown in Table 8-4. Table 8-4 is not exhaustive but is intended to highlight the diversity of failure modes experienced by several different classes of components.
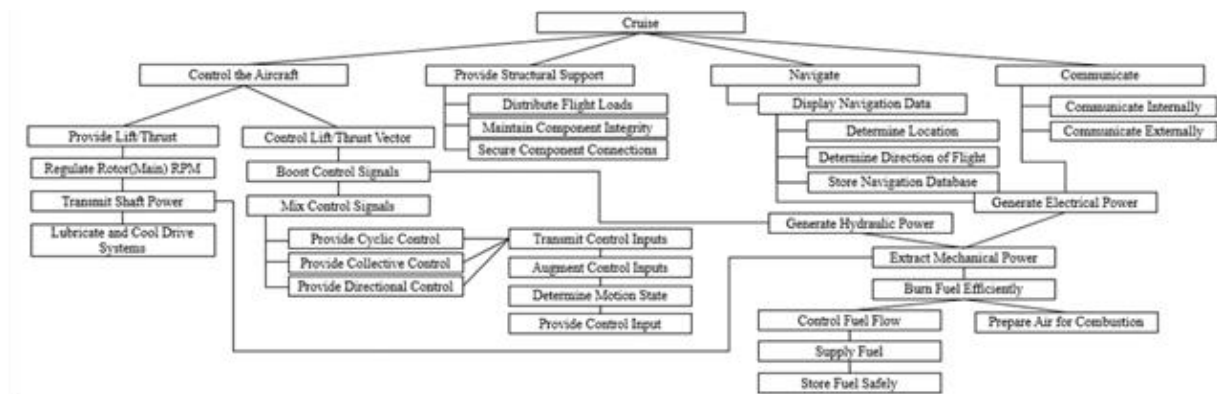
Figure 8-6. Notional Single Main Rotor Helicopter Functional Architecture for Cruise Segment
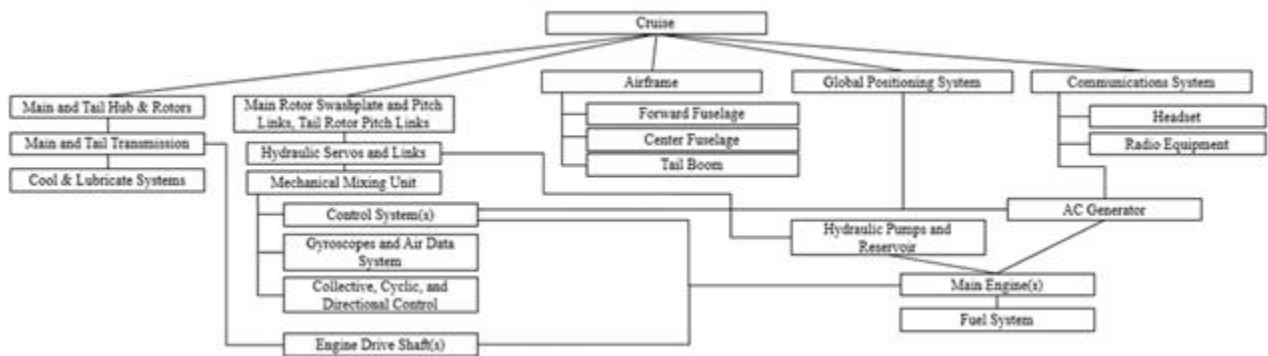
Figure 8-7. Notional Single Main Rotor Helicopter Physical Architecture Applicable to Cruise Segment

Figure content (fault tree diagram):

Cruise Mission Abort

- Engine Drive Shaft(s), Main Engine, Fuel System 1, Fuel System 2
- AC Generator, Global Positioning System, Communications System
- Tail Rotor Pitch Links, Tail Rotor, Tail Transmission
- Main Rotor Swashplate, Main Rotor Pitch Links, Main Rotor, Main Transmission, Oil System, Hydraulic Servos, Hydraulic Pump, Mixing Unit, Cyclic Control Inputs, Collective Control Inputs, Directional Control Inputs
- Forward Fuselage, Center Fuselage, Tail Boom

Cruise Safety Failure

- Engine Drive Shaft(s), Main Engine, (Fuel System 1, Fuel System 2)
- AC Generator
- Tail Rotor Pitch Links, Tail Rotor, Tail Transmission
- Main Rotor Swashplate, Main Rotor Pitch Links, Main Rotor, Main Transmission, Oil System, Mixing Unit, Cyclic Control Inputs, Collective Control Inputs, Directional Control Inputs
- Forward Fuselage, Center Fuselage, Tail Boom

Figure 8-8. Mission Abort and Safety Critical Fault Trees for Cruise Segment

Table 8-4. Failure Modes by Component Type.

Component Class	Failure Modes
Structural [58] [62]	Buckling, Corrosion, Delamination, Disbond, Cracking, Bolt Loosening, Hole Elongation, Foreign Object Damage
Electrical [63]	Loss of Power, Loss of Backup Power, Loss of Input Signal, Unable to Process/Accept Input Signal, Unable to Output Command Signal, Sensor Failure
Dynamic Component [58] [64] [65]	Bearing Failure, Damper Failure, Foreign Object Damage, Unbalanced Component, Misaligned Component, Component Structural Failure, Oil Leakage or Sludge Accretion
Actuation Component [58]	Loss of Power, Loss of Input Signal, Loss of Sensor for Control Feedback, Foreign Object Damage, Component Structural Failure, Leak or Broken Connection

For the failure modes experienced by each class of component, the repair time, cost, and severity may vary substantially. For example, loss of a blade erosion strip, is technically a failure within the main rotor system, but will not constitute a safety critical failure. For the representative data used for the notional helicopter example, it is assumed that the failures encountered allow the component to continue performing its function well enough to return to the point of origin, constituting a mission abort rather than the loss of the vehicle.

When component failure modes are diverse, it may be useful to model failure modes of individual components rather than the component as a whole. Because the current model is based on the definition of subsystems and components, which can easily be abstracted to a set of failure modes, it is possible for the model in its present state to handle the modeling of specific failure modes. However, for the notional helicopter modeled in this example for illustrative purposes, the decomposition to component failure modes is not performed.

8.3.2 Mission Modeling

The mission used for modeling this notional single main rotor helicopter is 110-minutes in duration and is intended to be representative of a common transport mission for a civil helicopter. The duration of each mission phase is based on a mission in an Army Tactical Environment and the details of the phased-mission are shown in Table 8-5. [66]

Table 8-5. Model Implementation Phased-Mission.

Phase	Duration (min)
Start-Up	5
Takeoff	2
Cruise-Out	45
Setdown	6
Cruise-Back	45
Setdown	2
Shutdown	5

8.3.3 Approximating Failure Densities

Without data from a regulatory agency, manufacturer or large operator of a given aircraft system, it is impossible to construct accurate distributions for the failure density and mean time to repair. However, because the discrete-event simulation is meant for use at the conceptual design level, and is intended to incorporate uncertainty through probabilistic inputs, accurate failure distributions are not required as long as the user accepts that the outputs of the simulation will only be representative.

The process of using data for a similar system is undertaken in this study for a generic helicopter. Given that this is a generic helicopter, no inference should be made regarding the reliability, availability, or maintainability to any current system. Because the data used to model the systems is notional, the data used is not presented in detail here. The failure densities are assumed to be Gaussian with a standard deviation equivalent to 10 percent of the mean value obtained for similar parts. Component cost to repair or replace data is difficult to estimate without a reliable source. Therefore, repair cost will not be discussed in the

implementation due to significant limitations regarding data availability. However, the simulation retains the capability to track the part cost for both failure-based and preventive maintenance.

In this implementation, replacement/repair windows for parts with specified service intervals (multiples of a required 25-hour inspection) are assumed to be 0 percent of the service interval in the field and 5 percent of the service interval if the aircraft is already in the shop. [67] The service intervals are unknown and must be approximated. The approach is to set the value at 10 percent less than the mean of the specified failure density for a component while rounding to the nearest 25-hour inspection interval. Mechanical subsystems are assumed to have field inspection and maintenance windows while monolithic components and electrical subsystems are not assumed to have these specified field inspection and maintenance windows.

8.4   Results

The vehicle architecture discussed above is modeled using the integrated discrete-event simulation environment with the input process streamlined using an Excel interface. Because the MTBF and MTTR values are distributions rather than deterministic values it is appropriate to run several Monte Carlo cycles to appropriately capture the operations and support characteristics for the system.

For this implementation, meant to show the capability of the developed discrete-event simulation, the data used is notional. This includes the models used to determine the component life, time to repair, shop availability, and logistical delay time. System cost is not modeled in this example. The key metrics tracked in this example include the operational availability, maintenance free operating period, the total number of failures of each component, the total number of preventive maintenance actions required for each component, and the maintenance man hours required for failure-based and preventive maintenance during the simulation.

The simulation used to evaluate the operational availability and mean time between failures is run for 5,000 hours and 200 Monte Carlo cycles. For the simulation used to evaluate the MFOP, 200 missions are attempted in each of the 200 Monte Carlo runs. During the evaluation of the maintenance free operation period, it is assumed that the vehicle has completed a maintenance recovery period prior to starting a maintenance free operation period, which means that all necessary inspections have been performed. Performing Monte Carlo cycles allows for quantification of the variability of the key metrics evaluated. This means that for a given reported value, there is also a confidence level associated with that value.

A histogram showing the distribution of the operational availability for each of the 200 Monte Carlo cycles is shown in Figure 8-9. This data can also be used to construct an inverse cumulative distribution that is

useful in visualizing the probability that the operational availability was above a given value in the simulations that were performed. For reference, lines are added on the inverse cumulative distribution that correspond to 75% and 95% confidence levels. The inverse cumulative distribution plot is shown in Figure 8-10.
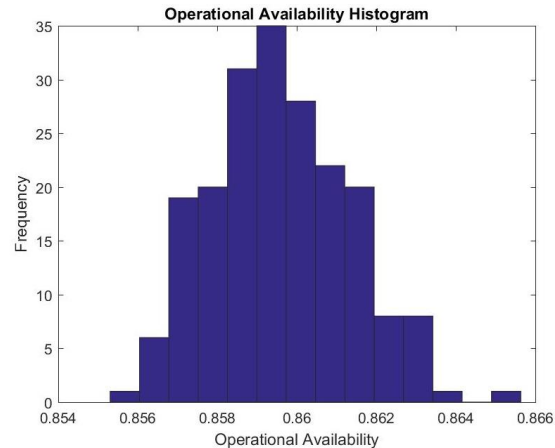
Figure 8-9. Operational Availability Histogram
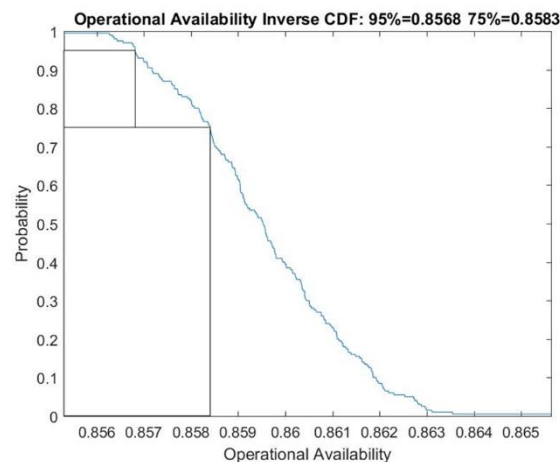
Figure 8-10. Operational Availability Inverse CDF

The Mean Time Between Failures (MTBF) is also determined during each Monte Carlo cycle. In the same way as the operational availability, the MTBF may be shown on a histogram, Figure 8-11, or used to construct an inverse cumulative distribution, Figure 8-12.
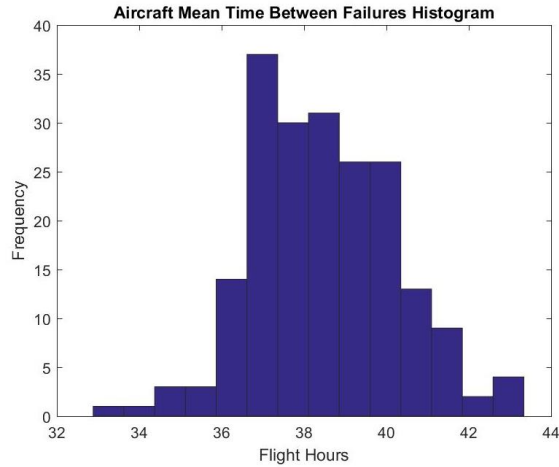
Figure 8-11. Mean Time Between Failures Histogram
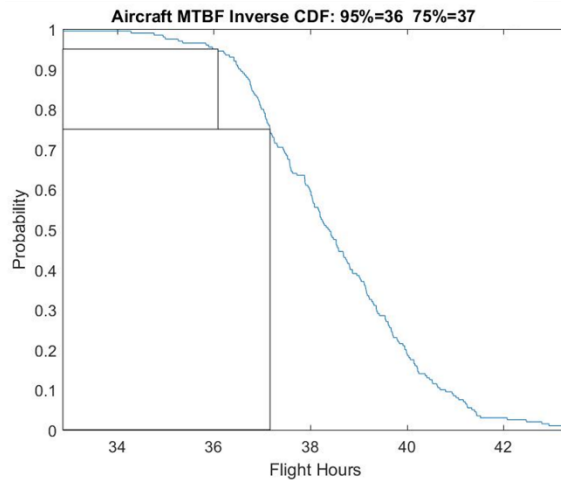
Figure 8-12. Aircraft MTBF Inverse Cumulative

During the 5,000-hour simulation, 291 hours were required to repair failed parts while 560 hours were required to repair parts that met condition-based failure criteria.

The results presented above show vehicle level metrics in detail but provide no insight into how much an individual subsystem or component is being serviced. By tracking the preventive maintenance performed on each component as well as failures experienced, it becomes possible to track components that fail most often, a valuable insight at the conceptual design level. Early in the design process this shows engineers where reliability improvements are the most needed. The average number of component failures during the 5,000-hour Monte Carlo cycles are shown in Figure 8-13 for the notional helicopter performing the transport mission shown in Table 8-5. During the same simulation, the number of repair actions performed on each component as a result of the component condition is shown in Figure 8-14.

Figure 8-13. Component Failures per Cycle

Figure 8-14. Condition-Based Actions per Cycle

The MFOP of the helicopter is evaluated next using a different mode of the integrated discrete-event simulation environment. Similar to the other vehicle-level metrics, performing Monte Carlo cycles allows for estimation of the probability that a vehicle will be able to complete a number of flight cycles without required maintenance. The inverse cumulative distribution for the number of maintenance free flight cycles performed is shown in Figure 8-15.

Figure 8-15. Inverse Cumulative for MFOP Cycle Success

From Figure 8-13 and Figure 8-14, it is evident that, for this notional helicopter, the hydraulic pump, tail rotor pitch links, tail rotor assembly, and the main rotor would benefit greatly from reliability improvements. To simulate such an improvement, the mean of the mean time between failures is increased by 10 percent to simulate the effect of reliability improvements at the component and subsystem level. Using this approach, the model may be used to quantify the impact of investments in component reliability, an RDT&E cost, on vehicle level O&S metrics. This is a similar approach to that used by Bhattacharya when modeling the relationship between investment in reliability and cost. [68] The realized improvement to the operational availability is shown in the operational availability inverse cumulative distribution in Figure 8-16 and the improvement to mean time between failures for the aircraft is shown in Figure 8-17. Both metrics show improvement compared to the baseline values reported previously.
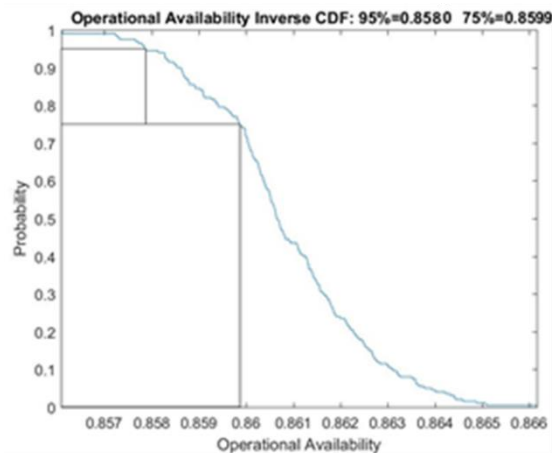
Figure 8-16. Improved Operational Avail. Inverse CDF

145
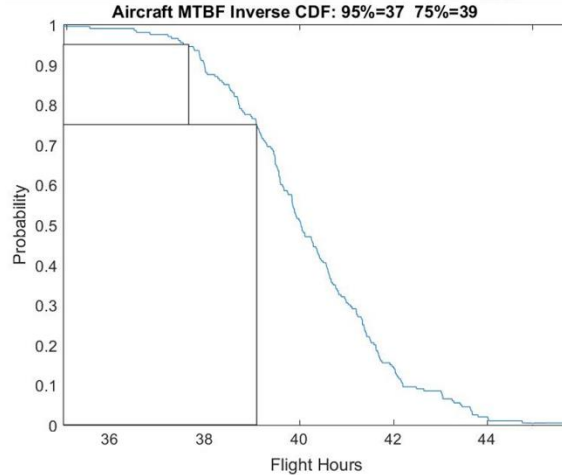
Figure 8-17. Improved Aircraft MTBF Inverse CDF

The effect of reliability improvements on the maintenance free operation period is shown in Figure 8-18. This shows that the maintenance free operation period is lengthened slightly due to reliability improvements.
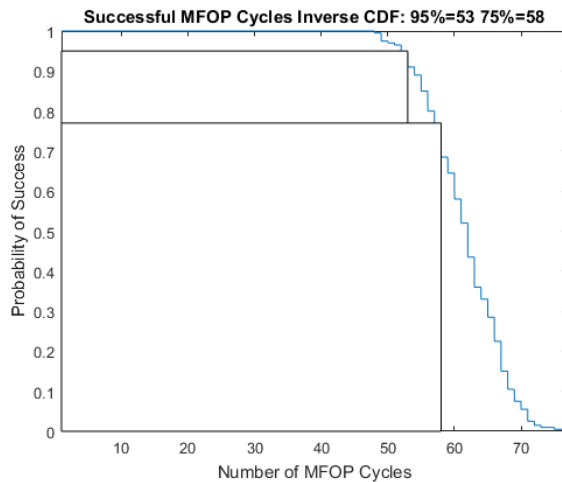
Figure 8-18. Improved Inverse CDF for MFOP Success

During the 5,000-hour simulation, 283 hours were required to repair failed parts while 563 hours were required to repair parts that met condition-based failure criteria.

The notional results shown above for a notional single main rotor helicopter seek to show the value of the integrated discrete-event simulation environment. At the conceptual design level, improvements to component reliability and maintainability are traditionally evaluated by subject matter experts in a qualitative manner. When the effect of multiple reliability improvement programs are compounded, with the possible side effect of longer repair times, it becomes difficult to accurately predict the effect on the vehicle level metrics. This implementation shows the ability to perform trades at the component level

146

against key operations and support metrics for the vehicle that are quickly evaluated quantitatively. Although not demonstrated here, it is possible to perform additional trades at the system architecture level by including component redundancy or the use of advanced concepts. Performing architectural trades results in modification of Figure 8-7 and Figure 8-8 to account for changes in the physical architecture and fault trees. The ability to rapidly evaluate the effect of adjusted architectures or changes in component reliability and maintainability are critical to assessing the impact of new technologies on a vehicle system.

8.5   Key Questions and Implications of Model Assumptions in FY17 Scope of Work

Understanding the limitations of a model is as important as understanding its realm of predictive capabilities. Many limitations and assumptions used in generating the model have been discussed in the preceding sections in great detail. Several questions asked by peers and attendees at the 73rd AHS forum are presented in Table 8-6 to highlight how specific limitations have been addressed in FY17 or are not currently planned to be addressed at present.

The first question regarding mission critical and non-mission critical event trees has been addressed by performing a functional and physical decomposition of the air vehicle of interest during FY17. In this case, the vehicle of interest is a single main rotor helicopter. Critical functions that are required during each individual mission phase are grouped and the components that contribute to this functionality are included in the event tree for that individual mission phase. In this way, a methodology is developed that is useful in performing similar studies for different classes of air vehicles.

Another common question is whether maintenance actions for individual components are scheduled or whether maintenance actions only occur when a failure occurs. On air vehicles, many components require periodic maintenance even in the absence of failures. This required effort, discussed here as preventive maintenance was added to the model in FY17. This is captured in the model by setting the scheduled maintenance interval for a part, which triggers required maintenance actions (but not a part failure) when the part life is with a user-defined range of that maintenance interval. This leads to the question of whether unscheduled and scheduled maintenance are tracked separately or whether the results are aggregated. In the FY17 work, the unscheduled and scheduled maintenance man hours and part repair cost are tracked separately.

Another relevant question is related to system redundancy. It is worth noting that there are 'hot' redundant systems that operate at a fully-operational state continuously, 'warm' redundant systems that operate continuously but at a reduced workload compared to the primary system, and 'cold' redundant systems that do not operate while the primary system is fully operational. Modeling 'hot' redundant systems is accomplished by adding the appropriate logic in the fault trees and including another identical system to

the simulation manually in the Excel interface. 'Warm' and 'cold' redundant systems cannot be modeled accurately at present because the parts will inherently age differently than the primary system because they operate at a lower workload than a 'hot' redundant system. Such non-uniform aging is not provided for in the current model.

The process used for part replacement and repair is another important topic. In the FY17 work, it is assumed that part repair and replacement fully reset the part age. However, this may not be the case for some components. As an example, a repaired or refurbished component may have a significantly shorter time between failures than a new part used to replace the failed component. Additional insight into each component would be required to effectively address this concern and modifications to the simulation would be required to incorporate this consideration.

Only a single level of maintenance is considered in the FY17 work. One significant expansion of this work would be to both include additional maintenance levels while also considering more than a single vehicle, thus allowing for fleet-wide considerations rather than concerns only at the vehicle level.

The final limitation discussed is related to vehicle safety. Addressing safety concerns requires significant analysis at the tails of the distributions because safety critical failures are rare. In the context of a single vehicle, the likelihood of a safety-critical failure on mission is small and does little to improve the prediction of the expected maintenance workload for a single vehicle. This is the case because the non-safety-critical failures occurring at a much higher frequency than safety critical failures constitute a large majority of the maintenance actions while typical safety-critical failures often lead to significant vehicle damage. Such damage cannot be accurately predicted at a high level due to the many factors that contribute to the level of damage sustained during a safety-critical event including but not limited to terrain, weather, load-out, fuel state, and the flight conditions when the failure occurred.

The purpose of this discussion is to identify key implications of assumptions in the current model and modeling efforts undertaken in FY17 to reduce these limitations and improve the predictive capability of the model. The discussion above is summarized in Table 8-6.

Table 8-6. Model Limitations and Questions Presented at the AHS 73rd Forum

Question/Limitation	Addressed in FY17	Not Addressed
1. How are the mission-critical and non-mission critical event trees populated?	A functional and physical decomposition of a notional Bell 206 helicopter is performed, grouping essential functions (and the related components) for each mission phase	
2. No maintenance actions are scheduled at regular intervals to maintain parts prior to the occurrence of a failure	Scheduled maintenance intervals are included that trigger maintenance actions when a part is within a given percentage of the service interval	
3. The maintenance man hours and component cost for unscheduled and preventive maintenance are not tracked separately	The model has been updated to track unscheduled and preventive maintenance metrics separately	
4. Are redundant systems considered?	Adding 'hot' redundant components can be completed manually simply by the addition of another identical system in the simulation	Including 'cold' and 'warm' component redundancy requires modification to the simulation approach as well as additional simulation inputs to age the parts differently than a 1-to-1 mapping with flight hours
5. Parts may be repaired rather than replaced during maintenance, how is the component life reset in this case?		It is assumed that both part repair and replacement fully reset part age, additional data and modeling are required to accurately capture this effect
6. Are only single level maintenance paradigms considered?		No multi-level maintenance paradigms are implemented in the FY 17 work
7. Safety critical failures, though rare, do occur. Are these types of failures considered in the simulation?		Looking at system safety requires examination of the extreme tails of complete failure distributions. Considering the non-safety critical failures where the vehicle can return to base for maintenance captures an overwhelming portion of the maintenance and operational activity of the vehicle

8.6   Opportunities for Future Work

The development and implementation of the integrated discrete-event simulation discussed in this paper provides many opportunities to improve the model in several key areas: component, mission, and maintenance modeling.

At present, components are tracked using the number of accrued flight hours. For some components, like landing gear or even some turbine engine components, tracking hours alone may not be sufficient to fully capture life limits and failure phenomenon. [69] Tracking the number of flight cycles in addition to flight hours would represent an improvement in the component modeling approach.

Part failure is modeled as a binary phenomenon though limits are set to trigger replacement at regular inspection intervals. This means that individual components show no signs of failure in the immediate time preceding failure. Given that component degradation prior to failure is often a gradual process, components do show signs of degradation prior to failure. This slow degradation was referenced in interviews conducted with pilots and operators who indicated that they inspect for fluid leakage or grease accumulation during pre-flight checks as a method to ensure mechanical systems are functioning properly. Appropriately modeling mechanical components, considering that part failure is not a binary phenomenon, requires additional data. Specifically, this data includes the length of time before failure when degradation becomes noticeable by a pilot or mechanic. Including this effect essentially involves adding condition-based maintenance in addition to the currently implemented time-based and failure-based maintenance. [70]

Another modeling consideration is how the age of a repaired part is reset. When a part is replaced, it is natural to reset the component age to zero. However, when a part is repaired in the field, it is likely that the part will not be restored to a like-new condition. This is based on the notion that a refurbished part likely has a reduced mean time between failures as compared to a brand new component. Capturing the effect of repair actions using empirically derived values would represent an improvement of the model and would decrease the optimism of the model in predicting component life after component repairs are complete.

A common technology implemented in an effort to improve the operational availability is Health and Usage Monitoring Systems (HUMS). HUMS monitors a large number of key parameters throughout the air vehicle to predict failures before they happen and alert maintenance personnel to act. The current approach uses a discrete event simulation and therefore abstracts away many of the details that would be required to fully model HUMS. However, the purpose of implementing a HUMS system is to predict impending failures so that they can be fixed or mitigated to reduce downtime. In this simulation, the part failure times are assigned a priori using a user-provided random distribution, which may provide a means to approach the problem successfully from a different perspective. By knowing the failure time, there may be a way to send the

vehicle to preventive maintenance before the failure occurs. However one must also model the fact that HUMS may not always warn of an impending failure and may lead to the replacement of parts that are not approaching a failure. The adaptation of the simulation to include HUMS should not be considered a trivial modeling task. Modeling HUMS is not easily accomplished using simple technology k-factors because HUMS fundamentally changes the approach used to perform maintenance rather than simply adjusting a maintenance parameter. Although implementing HUMS may seems like an obvious and purely beneficial approach to increase operational availability, there is definitely a tradeoff due to the increased number of analysts required to support HUMS on the air vehicle and more frequent part replacement as mentioned previously. Therefore, HUMS should not be arbitrarily included in vehicle systems without a full consideration of the associated costs and benefits. Including the analysis of HUMS in a modeling framework such as this may be one part of the effort required to determine the efficacy of including HUMS in future air vehicles.

Phased-mission modeling is another aspect of the present model. However, in the presence of a mission abort, it is assumed that the helicopter is nursed back to the point of origin for maintenance without additional age being accrued by the parts on the vehicle. In reality, parts are aged while returning to the point of origin. Beyond simply accruing flight hours, the functional components may accrue additional wear during the flight due to increased vibratory loads or increased strain on one system due to the failure of another. Modeling the increase in wear on one system due to the failure from another would require significant understanding of the physical system architecture and layout. This consideration, though relevant in the physical scenario, is likely beyond the scope of this high-level model intended to predict vehicle-level operations and support costs, operational availability, and MFOP.

Maintenance actions performed in the current model are assumed to occur at a single-level, one maintenance facility. Part availability is included using a user-specified scaling factor that multiplies the aircraft downtime. Improving the modeling approach to include multi-level maintenance paradigms as well as the delay (in hours) due to difficulty in acquiring certain parts would make the model significantly more realistic. However, this represents a significant increase in model complexity as modeling multiple maintenance facilities and the logistics network required to supply spare parts and tools requires significant knowledge about the maintenance program for a particular aircraft system and may not be the same between different aircraft of interest.

The shortcomings in the present model provide room for improvement in the component, mission, and maintenance modeling approach. However, even with these assumptions in the current model, it is possible to perform tradeoffs at the component and system architecture level to determine the effect of improved

component reliability or a different architecture on the reliability, availability, and maintainability of the system.

8.7   Available Applications for the Developed Model

Much effort is expended in documenting the limitations of the discrete-event simulation in its current state. Even with these limitations, it is possible to apply this simulation tool to evaluate different architectures with varied component reliabilities at the single vehicle level. There are three distinct approaches that are envisioned as opportunities to apply this simulation framework. The first is comparing the MFOP and operational availability for a variety of vehicle concepts and architectures using a standardized library of components. This relies on the individual architecture of the vehicles and the 'hot' redundancy within each as well as the criticality of different failures. The second approach would be to look at a single vehicle to determine how improvement or degradation to the reliability (MTBF) and/or maintainability (MTTR) of vehicle components affect the operational availability and MFOP for that single vehicle. The third option is to consider the first and second options together as a way to determine which vehicle architecture is the most robust to changes in the maintainability and reliability of individual components.

These three approaches provide a means to evaluate various concepts as part of an analysis of alternatives for example at the vehicle level. Although no provision is currently made for fleet-level modeling, improving the operational availability and MFOP of individual assets leads to improvements in fleet performance. As such, the same fleet-level performance may maintained with a reduced number of assets.

8.8   Considerations Regarding Reliability and Maintainability Data Needs

There are three significant limitations to applying this model to a current vehicle system. The first is more accurately understanding how maintenance is currently performed. The second is better understanding how component age resetting should be completed and the third is the raw maintenance data required to accurately populate failure distributions for each component.

Knowledge about how maintenance is currently performed on a particular aircraft is critical to accurately modeling the system. This knowledge includes not only the different levels of maintenance that are performed but also the type of maintenance actions that are performed at each level. These actions may include inspections, replacement of expendable items such as filters and seals, or the replacement of entire assemblies. Obtaining such knowledge requires a close relationship with operators and access to current maintenance documentation and practices.

Additional data needs are related to eliminating the assumption that refurbished components are 'as good as new.' This also requires a close relationship with the OEM to obtain reliability data for both new and

reconditioned parts needed for an accurate comparison. In addition to proprietary information protection another reason that this information may be difficult to obtain is because when used together could be used to conclude that refurbished parts may not be nearly as reliable as new components. Such a conclusion could be detrimental to various business interests. Although raw data may be difficult to obtain, there may be a consensus or rule of thumb within the industry that would allow for a reasonable approximation that could be used in the model that has yet to be documented.

Additional data required is used to populate the part reliability and maintainability distributions. This data includes raw part life, failure mode, failure impact on the mission, maintenance action (repair or replace), required repair/replace time, and repair/replace cost data for each tracked component on the helicopter. Several open source documents provide the mean time between failures or the mean repair time for a component. However, such knowledge about the mean of the distribution provides only limited insight into component reliability and maintainability because the distribution is abstracted to a single number that provides no insight on the actual distribution shape or width. Collecting such data again requires a close partnership with an OEM or operator and proprietary information agreements.

In the event that an operator or OEM makes the data available, condensing and operating on such data is a non-trivial task. Collecting the data outlined above will likely result in in many gigabytes/terabytes of non-uniform text documents that must be interrogated rapidly and efficiently to distill the metrics of interest that are required for this model, a task that can be classified as a 'big data' problem. Alone, the data reduction is a task within itself that is often undervalued because the work products of such a task provide only an intermediate result that must be further applied within other simulations. Considering this, only collecting the data from an OEM or operator is not enough, steps must be taken to reduce the data to a usable form.

Given that there are significant challenges to both obtaining access and reducing the necessary component and system architecture data, it is worth searching for alternatives to this approach. Such alternatives may provide a more direct modeling approach that is not based on the component reliability data. However, such approaches must pass through verification and validation activities to ensure that the model is internally correct and represents the reality that we wish to capture.

The challenges of acquiring data to use in the developed discrete-event simulation are certainly formidable. Considering the full depth of the challenge of acquiring component reliability and maintainability data is necessary to provide a true appreciation for the work that remains to make a model such as this one useful in accurately predicting operational availability and MFOP for current air assets. Predicting operational

availability and MFOP for future assets must rely on data for components of current assets with adjustments made for increases or decreases in the reliability or maintainability a result of technology infusion or improved manufacturing processes. The data required is neither simple to acquire or reduce and motivates further exploration into alternative approaches not based upon component reliability and maintainability data.

8.9 Concluding Remarks

The methodology used to construct an integrated discrete-event simulation for evaluation of rotorcraft operations and support metrics has been presented. Specifically, this model may be used to estimate the Operational Availability, MFOP, mean time between mission affecting failures, part failure rates, and maintenance man hours required for both conventional and novel rotorcraft architectures. Using an integrated discrete-event simulation environment allows for quantitative trade studies at the subsystem or component level and the system architecture level against key operations and supportability metrics. The need for such an environment is highlighted in the implementation provided. Significantly improving components with the greatest number of maintenance actions required only slightly improved the operations and supportability metrics for the single vehicle considered.

One key limitation of the model is that the data requirements to accurately model a proposed system are immense. Because this environment is meant for use at the conceptual design level, part failure and repair time data are rarely available for the specific components of interest. Using data for similar systems provides a rough estimate of component and subsystem failure data, which may be used to calculate the operations and supportability metrics. Although this approach will not yield a value for any one of the metrics that is a true prediction for the realized system, it enables a quantitative evaluation of the system early in the design process among other vehicles modeled using this approach. Under this standardized approach each component must have representative failure and repair data that can be perturbed according to anticipated improvement or degradation.

Improvements in the current model are most beneficial in several key areas including component definition, mission representation, and maintenance modeling. Nonetheless, the model in its current state provides a framework to quantitatively evaluate operations and support metrics in the context of rotorcraft conceptual design activities.

9. References

[1] J. Arruda, A. Gavrilovski, H. Chae, E. Spero and D. Mavris, "The Capability Assessment and Tradeoff Environment (CATE) for Advanced Aerospace Vehicles and Technology Assessment.," Proceida Computer Science, 2014.

[2] J. R. Kizer, "Aircraft Conceptual Design Enabled by a Set-Based Approach for the Exploration and Bounding of Non-Hypercubic Design Spaces," Georgia Tech, Atlanta, 2016.

[3] T. Stephens, "Titanic: Getting Started with R - Part 5: Random Forests," 18 Jan 2014. [Online].

[4] L. Brieman, A. Cutler, A. Liaw and M. Wiener, "Brieman and Cutler's Random Forests for Classification and Regression," Oct 2015. [Online]. Available: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

[5] M. R. Kirby, D. N. Mavris and M. A. Largent, "A Process for Tracking and Assessing Emerging Technology Development Programs for Resource Allocation," in *1st AIAA Aircraft Technology, Integration, and Operations Forum*, Los Angeles, CA, 2001.

[6] K. Collins and D. N. Mavris, "Mobility Research for Future Vehicles: A Methodology to Create a Unified Trade-Off Environment for Advanced Aerospace Vehicles Fiscal Year 2015 Annual Report," Georgia Tech Aerospace Systems Design Lab, 2015.

[7] M. Kirby and D. Mavris, "An Approach for the Intelligent Assessment of Future Technology Portfolios," 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 2001.

[8] T. J. Gordon and Glenn, J. C., "Futures Research Methodology, Version 2.0Millennium Project of the," American Council for the United Nations University, 2003.

[9] Firat, A., W. Woon and S. Madnick, "Technology Forecasting - A Review," MIT Composite Information Systems Laboratory, Cambridge, 2008.

[10] R. Levary and D. Han, "Choosing a Technological Forecasting Method," *Industrial Management,* p. 14, 1995.

[11] J. Phillips, T. Heidrick and I. Potter, "Technology Futures Analysis Methodologies for Sustainable Energy Technologies," IEEE, 2005.

[12] J. P. Martino, Technological Forecasting for Decision Making, Amsterdam: North-Holland Publishing, 1993.

[13] A. Floyd, A Methodology for Trend Forecasting of Figures of Merit, Englewood Cliffs: Prentice-Hall, 1968.

[14] G. Birrocco, "ASSP TDA Status Brief for Future Vertical Lift S&T Team," 2009.

[15] A. Smith, K. Collins and D. Mavris, "Survey of Tecnology Forecasting Techniques for Complex Systems," in *AIAA SciTech Forum*, Grapevine, TX, 2017.

[16] G. Intepe, E. Bozdag and T. Koc, "The Selection of Technology Forecasting Methods Using a Multi-Criters Interval-Valued Intuitionistic Fuzzy Group Decision Making Approach," *Computers & Industrial Engineering,* pp. Vol 65 177-285, 2013.

[17] A. Cheng, C. J. Chen and C. Y. Chen, "A Fuzzy Multiple Criteria Comparison of Technology Forecasting Methods for Predicting the New Materials Development," *Technological Forecasting & Social Change,* pp. Vol 75 131-141, 2008.

[18] S. Mishra, S. G. Deshmukh and P. Vrat, "Matching Technological Forecasting Technique to a Technology," *Technological Forecasting & Social Change,* p. Vol 69, 2002.

[19] General Electrc, "The T901 Turboshaft Engine | GE Aviation," 2016. [Online]. Available: Geaviation.com.

[20] J. Leishman, Prinicples of Helicopter Aerodynamics, New York: Cambridge University Press, 2006.

[21] "DoD Award ITEP Advanced Helicopter Engine Contracts," 2017. [Online]. Available: http://www.janes.com/article/63148/dod-awards-itep-advanced-helicopter-engine-contracts.

[22] W. Yao, X. Chen, W. Luo, M. Van Tooren and J. Guo, "Review of Uncertainty-Based Multidisciplinary Design Optimization Methods for Aerospace Vehicles," *Progress in Aerospace Studies,* pp. Vol 47 Issue 6 450-479, 2011.

[23] R. Dybyad, "Helicopter Gross Weight and Center of Gravity Measurement System," AD-771 955, 1973.

[24] R. Thornburgh, "Continuous Trailing-Edge Flap for Helicopter Rotor Blades," 2012.

[25] B. Glaz, "Modeling of Nanosecond Pulsed Plasma Based Separated Flow Control," 2012.

[26] M. E. E. L. M. Barnett, "UH-60M Technology Readiness Level Assessment," 2001.

[27] Ames, "Future Directions in Rotorcraft Technology at Ames Research Center," American Helocopter Society, 2000.

[28] R. Halbig, "Ceramic Matrix Composites for Rotorcraft Engine," 2011.

[29] U. Aerospace, "Health and Usage Management Systems (HUMS)," 2017. [Online]. Available: http://utcaerospacesystems.com/cap/products/pages/health-usage-management-systems.aspx. [Accessed 2017].

[30] F. Gandhi, ""Reconfigurable Vertical Lift", AE Presents Lectures Series at Georgia Institute of Technology, November 2017".

[31] Bell Helicopter Textron, "Bell 206L4 Product Specifications," 2016.

[32] T. S. a. B. Maundrill, "Heli-Expo 2015: Introducing the H160," Shephard Media, 3 March 2015. [Online]. Available: https://www.shephardmedia.com/news/rotorhub/heli-expo-2015-introducing-h160/. [Accessed 1 November 2017].

[33] R. Kuhn, "Review of Basic V/STOL Aerodynamics, NASA TN D-733, March 1961".

[34] W. a. T. T. Stepniewski, "Open Airscrew VTOL Concepts, NASA Contractor Report 177603, September 1992".

[35] W. a. T. T. Stepniewsky, "Open Airscrew VTOL Concepts," NASA, 1992.

[36] E. D. Bouchard, M. Schmit, K. Collins and D. Mavris, "A Numerical Method to Calibrate and Forecast Technology Improvements for the UH-60 Helicopter Using NDARC," in *American Helicopter Society*, Fort Worth, TX, 2017.

[37] W. Johnson, "NDARC - NASA Design and Analysis of Rotorcraft Validation and Demonstration," in *merican Helicopter Society Aeromechanics Specialists' Conference*, San Francisco, 2010.

[38] W. Johnson, "NDARC NASA Design and Analysis of Rotorcraft Theory, Release 1.9," NASA Ames Research Center, Moffett Field, 2015.

[39] S. Ashok, A. Sirirojvisuth and A. Smith, "Closing the Gap between Capability and Affordability for System Upgrades," in *AHS 72nd Annual Forum*, West Palm Beach, 2016.

[40] K. B. Hilbert, "A Mathematical Model of the UH-60 Helicopter," National Aeronautics and Space Administration, Moffet Field, 1984.

[41] "Operator's Manual for UH-60A Helicopter, UH-60L Helicopter, EH-50A Helicopter," Washington, 2009.

[42] W. G. B. W. J. Hyeonsoo Yeo, "Performance Analysis of a Utility Helicopter with Standard and Advanced Rotors," in *American Helicopter Society Aerodynamics, Acoustics, and Test and Evaluation Technical Specialist Meeting*, San Frnacisco, 2002.

[43] GE Aviation, "T700-401C/-701C," Cincinnati.

[44] Mark W. Nixon, "Preliminary Structural Design of Composite Main Rotor Blades for Minimum," NASA, 1987.

[45] R. D. Leoni, Black Hawk The history of a World Class Helicopter, Reston: American Institute of Aeronautics and Astronautics, Inc., 2007.

[46] J. Price, S. Ashok, R. Armstrong, K. Collins, D. Mavris and D. Schrage, "Integrated Discrete Event Simulation Environemnt for Analysis of Rotorcraft Reliability, Availability, and Maintainability," in *American Helicopter Society*, Fort Woth, TX, 2017.

[47] American Helicopter Society, "Future Vertical Lift," 2016. [Online]. Available: https://vtol.org/what-we-do/advocacy/future-vertical-lift.

[48] Office of the Secretary of Defense, "Department of Defense Reliability, Availability, Maintainability, and Cost Rationale Report Manual," 2009.

[49] R. Armstrong, Z. Ernst, K. Collins and D. Mavris, "Reconfigurable Discrete Event Simulation of Rotorcraft Maintenance and Operations for Technology Assessment and Tradeoff," in *American Helicopter Society 72nd Annual Forum*, West Palm Beach, FL, 2016.

[50] Office of the Secretary of Defense, "Cost Assessment and Program Evaluation, Operating and Support Cost-Estimating Guide," 2014.

[51] A. Velden, D. Schrage, D. Arterburn, A. Smith, A. Sirirojvisuth and S. Ashok, "Probabilistic Certificate of Correctness for Helicopter Variants," in *Rotorcraft Virtual Engineering Conference Proceedings*, Liverpool, England, 2016.

[52] M. Gorac and M. J. Kwinn, "Lead the Fleet: Transitioning Army Aviation Maintenance From a Time Based System to a Usage Based System," Operations Research Center of Excellence, United States Military Academy West Point, 2004.

[53] S. P. Chew, S. J. Dunnett and J. D. Andrews, "Phased-Mission Modeling of Systems with Maintenancce-Free Operating Periods Using Simulated Petri Nets," *Reliability Engineering and System Safety,* pp. 980-994, 2008.

[54] P. Mitchell, "Maintenance-Free and Failure-Free Operating Periods to Improve Overall System Availability and Reliability," in *Design for Low Cost Operation and Support Proceedings*, Ottowa, 1999.

[55] Defense Acquisition University, "Mean Logistics Delay Time," 2016. [Online].

[56] G. Pryor, "Methodology for Estimation of Operational Availability as Applied to Military Systems," *ITEA Journal,* pp. Vol. 29 420-428, 2008.

[57] Chief of Naval Operations, "Operational Availability of Equipment and Weapons Systems -- OpNAV Instruction 3000.12A," 2003.

[58] J. J. Dougherty and L. D. Barrett, "Research Requirements to Improve Reliability of Civil Helicopters," NASA CR-145335.

[59] SAE International, "Aerospace Recommended Practice 4761," 1996.

[60] Bell Helicopter, "Operators Manual Army Model OH-58A/C," TM 55-1520-228-10, 1989.

[61] United States Department of Transportation, "Helicopter Flying Handbook," Federal Highway Administration -- Flight Standards Service, 2012.

[62] J. Sen and R. Everett, "Structural Integrity and Aging-Related Issues for Helicopters," ADP 010436, 2000.

[63] D. Galler and G. Slenski, *Causes of Aircraft Electrical Failures,* IEEE AES Systems Magazine, 1991.

[64] C. Bowen, L. Dyson and R. Walker, "Mode of Failure Investigations of Helicopter Transmissions," AD 881610, 1971.

[65] R. Vaughn, "U.S. Army Aviation Helicopter Dynamic Components," in *Understanding and Combating Aging*, Phoenix, AZ, 2009.

[66] COBRO Corporation, "Input Data, ARMS Model Simulation of the OH-58A in an Army Tactical Environment," ADA 041631, 1977.

[67] Civil Aviation Authority of New Zealand, "Airworthiness Directive Schedule -- Helicopters -- Bell 206B and 206L Series, and Augusta AB206 Series," 2016.

[68] S. Bhattacharya, V. Nagaraju and L. Fiondella, "Rotorcraft Tradespace Exploration Incorporating Reliability Engineering," in *American Helicopter Society 71st Annual Forum Proceedings*, Virginia Beach, VA, 2015.

[69] R. Weiser and M. J. Kwinn, "Managing Helicopter Maintenance Risk," Jet Support Services, Inc., 2015.

[70] U. D. Kumar, J. Crocker, J. Knezevic and M. El-Haram, Reliability, Maintenance and Logistic Support -- A Lifecycle Approach, Boston: Kluwer Academic, 2000.

[71] A. G. B. A. H.-G. C. E. S. D. M. James Arruda, "The Capability Assessment and Tradeoff Environment (CATE) for Advanced Aerospace Vehicle and Technology Assessment," in *Conference on Systems Engineering Research (CSER) 2014*, Redondo Beach, 2014.

[72] GE Aviation, "T700-701D turboshaft engines," Cincinnati.

[73] Department of the Army, "Operators Manual for UH-60A Helicopter, UH-60L Helicopter, EH-60A Helicopter," Washington, 1996.

[74] J. Mullins and S. Mahadevan, "Bayesian Uncertainty Integration for Model Calibration, Validation, and Prediction".

[75] M. Hirschberg, "Electric VTOL Wheel of Fortune, Vertiflite Magazine, March/April 2017, Also, www.vtol.org; eVTOL News".

10. Appendix

10.1 Appendix A: Test Objective Function for SeBBAS

10.1.1 Script to Call SeBBAS

```
%% Script to call and test SeBBAS code

clc; clear; close all;
% Constraint: A number corresponding to the constraint on the design space that will be applied.
% The constraints are numbered as follows. If a constraint is a negative number, than the inverse
%     of the constraint will be applied
%        1) Infeasible cross in middle, creating 4 squares in each corner
%        2) Upper triangle of hypercube is infeasible
%        3) Parabolic constraint
%        4) Points that fall within circle centered at (2.5, 2.5) are infeasible
%        5) Hyperbolic constraint
%        6) Triangular constraint in the middle of the hypercubic design space
%        7) linear band with slope 1 across hypercubic design space is infeasible
%        8) Cubic constraint on x1 variable
%        9) Slightly different cubic constraint
%        10) 3D sphere

% Setting constraint number
Constraint = 1;

% Setting the run_Case name and the total budget available for this study
run_Case = ['Test_C', num2str(Constraint)];
total_Budget = [1000, 3000, 10000];

% Writing the DV_Range_Filename file
DV_Range_Filename = 'Test_DOE_DV_Range.csv';
DV_Range_path = [pwd, '/R Files/Design Variable Range/', DV_Range_Filename];

% Setting up lower and upper bounds for the design variables and writing the table to the proper
% file location which was shown above
x1 = [0; 5];
x2 = [-5; 5];
DV_Range = table(x1, x2);
% x3 = [-5; 5];
% DV_Range = table(x1, x2, x3);
writetable(DV_Range, DV_Range_path)

% Defining the other parameters required
validation_Case = 50000;
max_Iter = 8;
tolerance = 0.01;

% Running SeBBAS
[suggested_DOE] = SeBBAS(run_Case, total_Budget, DV_Range_Filename, validation_Case, max_Iter, tolerance, ...
@run_Test_SeBBAS, Constraint);

% Plotting the results of the classification of SeBBAS
ind_s = find(suggested_DOE.Classification == 1);
ind_f = find(suggested_DOE.Classification == -1);

% Plotting for either 2D or 3D cases the number of correct classifications and incorrect
% classifications
if abs(Constraint) == 10
    x1 = suggested_DOE.x1;
```

159

```
    x2 = suggested_DOE.x2;
    x3 = suggested_DOE.x3;

    figure(1)
    plot3(x1(ind_s), x2(ind_s), x3(ind_s),'go')
    xlabel('x1')
    ylabel('x2')
    zlabel('x3')
    axis([-5, 5, -5, 5, -5, 5])
else
    x1_Final = suggested_DOE.x1;
    x2_Final = suggested_DOE.x2;

    figure(1)
    plot(x1_Final(ind_s), x2_Final(ind_s),'go', x1_Final(ind_f), x2_Final(ind_f), 'ro')
    xlabel('x1')
    ylabel('x2')
    axis([x1(1), x1(2), x2(1), x2(2)])
    legend('Correct Classifications', 'False Positive Classifications')
end
```

10.1.2 Test Objective Function

```
function [Classification] = run_Test_SeBBAS(DOE, Constraint)
% This function runs a series applies various constraints to create a non-hypercubic design space to
% test the accuracy of the SeBBAS.m function. Most of the functions are 2D, but can be expanded to
% any dimensions
%
% Inputs:
%   DOE: NxM array containing a DOE or set of design points to run through the constraint. The columns
%        of the array correspond to the M design variables, and the columns must be in the same order
%        as the variables listed in the VariableNames cell array. Rows of the array correspond to
%        differentn design points
%   VariableNames: Mx1 cell array containing strings pertaining to each design variable.
%   Constraint: A number corresponding to the constraint on the design space that will be applied.
%        The constraints are numbered as follows. If a constraint is a negative number, than the inverse
%        of the constraint will be applied
%           1) Infeasible cross in middle, creating 4 squares in each corner
%           2) Upper triangle of hypercube is infeasible
%           3) Parabolic constraint
%           4) Points that fall within circle centered at (2.5, 2.5) are infeasible
%           5) Hyperbolic constraint
%           6) Triangular constraint in the middle of the hypercubic design space
%           7) linear band with slope 1 across hypercubic design space is infeasible
%           8) Cubic constraint on x1 variable
%           9) Slightly different cubic constraint
%           10) 3D sphere
%
% Outputs:
%   Classification: A Nx1 array that contains either a (1) or (-1) for each design point, with a
% value of 1 corresponding to a feasible design point, and value of -1 meaning infeasible

% Initializing run settings
failedCaseTotal = 0;

% Calculating the number of cases in the array
numCases = length(DOE(:,1));

% Initializing feasibleCase array
Classification = zeros(numCases, 1);
```

```matlab
% Calling constraint function to classify DOE
for k = 1:numCases
    % Running NDARC and processing results
    [Classification(k)] = applyConstraint(DOE(k,:), Constraint);

end

%===================================================================================
============================

function [Classification] = applyConstraint(DesignVariables, Constraint)
% This function applies the desired constraint to the current design point
%
% Inputs:
%   DesignVariables: MxN array containing the DOE for N design variables, where M = numLHcases + numRandomCases
%   Constraint: A number corresponding to the constraint on the design space that will be applied.
%      The constraints are numbered as follows. If a constraint is a negative number, than the inverse
%      of the constraint will be applied
%          1) Infeasible cross in middle, creating 4 squares in each corner
%          2) Upper triangle of hypercube is infeasible
%          3) Parabolic constraint
%          4) Points that fall within circle centered at (2.5, 2.5) are infeasible
%          5) Hyperbolic constraint
%          6) Triangular constraint in the middle of the hypercubic design space
%          7) linear band with slope 1 across hypercubic design space is infeasible
%          8) Cubic constraint on x1 variable
%          9) SLightly different cubic constraint
%          10) 3D sphere
%
% Outputs:
%   Classification: -1 for infeasible design, 1 for feasible design

x1 = DesignVariables(1);
x2 = DesignVariables(2);

sign_Constraint = sign(Constraint);
Constraint = abs(Constraint);
if Constraint == 1
    if (x1>2 && x1 <3) || (x2>2 && x2<3)
        Classification = -1;
    else
        Classification = 1;
    end
elseif Constraint == 2
    if x1+x2>5
        Classification = -1;
    else
        Classification = 1;
    end
elseif Constraint == 3
    if x2 > (x1-2.5)^2 + 0.5
        Classification = -1;
    else
        Classification = 1;
    end
elseif Constraint == 4
    if (x2-2.5)^2 + (x1-2.5)^2 < 0.5
        Classification = -1;
    else
        Classification = 1;
    end
```

```matlab
elseif Constraint == 5
    if (x2-2.5)^2 -(x1-2.5)^2 < 1
        Classification = 1;
    else
        Classification = -1;
    end
elseif Constraint == 6
    if x2 > 1 && x2 < 1 + x1 && x2 < 6-x1
        Classification = -1;
    else
        Classification = 1;
    end
elseif Constraint == 7
    if x2 > x1 - 1 && x2 < x1 + 1
        Classification = -1;
    else
        Classification = 1;
    end
elseif Constraint == 8
    if x2 > (x1-2.5)^3/3.125
        Classification = -1;
    else
        Classification = 1;
    end
elseif Constraint == 9
    if x2 > (x1-2.5)^3/3.125 && x2 < -(x1-4)^3/3.125
        Classification = -1;
    else
        Classification = 1;
    end
else
    x3 = DesignVariables(3);

    if 4 < x1^2 + x2^2 + x3^2
        Classification = -1;
    else
        Classification = 1;
    end
end

% Taking inverse of classification if the constraint value is negative
Classification = Classification * sign_Constraint;
```

10.2   Appendix B: Current NDARC Design Variables Available

Table 10-1: NDARC variables available in the Aircraft Configuration template file

NDARC Variable	Description
diskload	Disk loading
CWs	Blade Loading (C_W/σ)
CD_fus	Zero lift drag coefficient of fuselage
CD_fit	Drag coefficient for fixtures and fittings
CDV_fus	Vertical drag coefficient of fuselage
CD_MR_hub	Drag coefficient of the main rotor hub
CD_MR_pylon	Drag coefficient of the main rotor pylon
CD_TR_hub	Drag coefficient to the tail rotor hub
log_cg_XoL	Location of the cg in non-dimensional x-coordinate

Vtip_ref	Reference tip speed
TECH_body	Technology factor for basic body
TECH_blade	Technology factor for blade weight
TECH_gb	Technology factor for gear box weight
TECH_rs	Technology factor for rotor shaft weight
TECH_ds	Technology factor for drive shaft weight
TECH_eng	Technology factor for engine weight
TECH_drag_MR	Technology factor for drag of main rotor
TECH_drag_TR	Technology factor for drag of tail rotor
TECH_RWfc_b	Technology factor of boosted rotary wing flight control weight
TECH_RWfc_mb	Technology factor of control boost mechanism of rotary wing flight control weight
TECH_RWfc_nb	Technology factor of non-boosted rotary wing flight control weight
TECH_RWhyd	Technology factor for weight of rotary wing flight control hydraulics
TECH_cost_maint	Technology factor for maintenance cost
Wcrew	Weight of the crew
Wpay	Payload weight
Peng	Engine power (SLS static takeoff rating)
Pacc_0	Accessory power loss constant
Ki_prop	Axial cruise propeller induced velocity factor

Table 10-2: NDARC variables in the engine template file

NDARC Variable	Description
SP0C_tech	Technology factor for specific power at MCP
sfc0C_tech	Technology factor for specific fuel consumption at MCP

Table 10-3: NDARC variables for the sizing conditions input file

NDARC Variable	Description
VROC	Vertical rate of climb flight speed
VROC_alt	Reference altitude for rate of climb sizing condition
VROC_temp	Reference temperature for rate of climb sizing condition
Vkts	Horizontal flight speed velocity
VFWD_alt	Reference altitude for horizontal flight sizing condition
VFWD_temp	Reference temperature for horizontal flight sizing condition

Table 10-4: NDARC variables for the mission input file

NDARC Variable	Description
FT_hov_time	Time spent in hover for fuel tank sizing
MIS_hov_time	Time spent in hover mission segment
MIS_hov_alt	Reference altitude for hover mission segment
MIS_hov_temp	Reference temperature for hover mission segment
FT_cr_time	Time spent in cruise for fuel tank sizing
MIS_cr_Vkts	Forward flight velocity in cruise mission segment
MIS_cr_time	Time spent in cruise mission segment

MIS_cr_alt	Reference altitude for cruise mission segment
MIS_cr_temp	Reference temperature for cruise mission segment
MIS_res_alt	Reference altitude for reserve mission segment
MIS_res_temp	Reference temperature for reserve mission segment
MIS_res_Vkts	Forward flight velocity in reserve mission segment
MIS_idle_alt	Reference altitude for idle mission segment
MIS_idle_temp	Reference temperature for idle mission segment

10.3   Appendix C: Sample "NDARC Calibration Settings.inp" File&RUN_SETTINGS

```
SaveAsFilename = Single Input File 2                   Xprop = 4
   NumRuns =  1                                        CTs_sep = .04, .1, .01
   inducedPopulationFactor =  3                        dsep = 1.6
   inducedGenerationFactor =  6                        Xsep = 2.9
   profilePopulationFactor =  10                       df1 = 0
   profileGenerationFactor =  6                        df2 = 0
&END                                                   Xf = 1.3
                                                       MODEL_stall = 1
&INDUCED_NDARC_VARIABLES                               nstall = 10
   MODEL_ind = 2                                       fstall = 1
   Ki_hover = 1, 3, .1                                 dstall1 = 2.6
   Ki_climb = 1.125                                    dstall2 = 60
   Ki_prop = 2                                         Xstall1 = 2.2
   Ki_edge = 1, 2, .1                                  Xstall2 = 2.7
   CTs_Hind = .05, .15, .01                            do1 = .2
   kh1 = 0, 2, .1                                      do2 = 4.7
   kh2 = 0, 1, .1                                      dsa = 0
   Xh2 = 2                                             MODEL_comp = 1
   CTs_Pind = .05, .15, .001                           fSim = 1
   kp1 = 1.25                                          thick_tip = .08
   kp2 = 0                                             dm1 = .01
   Xp2 = 2                                             dm2 = .79
   kpa = 0                                             Xm = 3.1
   Xpa = 2                                             Mdd0 = 0, 2, .01
   ko1 = 0                                             Mddcl = 0, 2, .01
   ko2 = 8                                          &END
   Maxial = 1.176
   Xaxial = .65                                     &CALIBRATION_DATA_SET
   mu_prop = 1.59                                      #  mu muz CT/s MAT Offset Actual_Kappa Actual_Cd
   ka1 = 0                                             0, 0, .05997, .6163, 0, 1.0158, .00837
   ka2 = 0                                             0, 0, .06894, .6163, 0, 1.0549, .00842
   ka3 = .92                                           0, 0, .07814, .6163, 0, 1.0887, .0085
   Xa = 5                                              0, 0, .08744, .6163, 0, 1.1206, .00867
   mu_edge = .25, .4, .01                              0, 0, .0969, .6163, 0, 1.1486, .00888
   ke1 = .5, 1.5, .01                                  0, 0, .10656, .6163, 0, 1.1724, .00916
   ke2 = 0, 2, .1                                      0, 0, .11641, .6163, 0, 1.1922, .00963
   ke3 = 1.4                                           0, 0, .12644, .6163, 0, 1.2064, .01075
   Xe = 2, 6, .1                                       0, 0, .13666, .6163, 0, 1.2172, .01196
   kea = 0                                             0, 0, .14685, .6163, 0, 1.228, .0134
   Ki_min = 1.085                                      0, 0, .15688, .6163, 0, 1.2364, .0152
   Ki_max = 10                                         0, 0, .16665, .6163, 0, 1.2408, .01786
&END                                                   0, 0, .1761, .6163, 0, 1.2431, .02135
                                                       0, 0, .1761, .6163, 0, 1.2431, .02135
&PROFILE_NDARC_VARIABLES                               0, 0, .19258, .6163, 0, 1.2353, .035
   TECH_drag = 1                                       0, 0, .1992, .6163, 0, 1.2323, .04284
   Re_ref = 0                                          0, 0, .20479, .6163, 0, 1.2289, .05138
   MODEL_basic = 2                                     0, 0, .20981, .6163, 0, 1.2225, .06283
   ncd = 24                                            0, 0, .21415, .6163, 0, 1.2181, .07734
   CTs_Dmin = .025, .1, .001                           .1687, 0, .06998, .7203, 0, 1.0706, .00857
   d0_hel = .001, .1, .001                             .1927, 0, .06997, .7351, 0, 1.1432, .00859
   d0_prop = 0, .01, .001                              .2167, 0, .07002, .7499, 0, 1.1412, .00859
   d1_hel = 0                                          .2407, 0, .06994, .7647, 0, 1.3486, .00865
   d1_prop = .1                                        .2645, 0, .07, .7794, 0, 1.4106, .00869
   d2_hel = .25, .75, .01                              .2883, 0, .0699, .7941, 0, 1.6112, .00869
   d2_prop = .25, .75, .01                             .312, 0, .06997, .8088, 0, 1.8293, .00884
   dprop = 2                                           .3354, 0, .07008, .8233, 0, 2.0511, .0091
```

164

.3587, 0, .06997, .8378, 0, 2.3868, .00936   .1928, 0, .08007, .7351, 0, 1.1503, .00866
.3816, 0, .07, .8522, 0, 2.7063, .00977   .2168, 0, .07994, .7499, 0, 1.2454, .00867
.4041, 0, .07009, .8664, 0, 3.2166, .0104   .2408, 0, .07997, .7647, 0, 1.2312, .00874
.4262, 0, .07003, .8804, 0, 3.8686, .01126   .2647, 0, .07997, .7795, 0, 1.3395, .00873
.4474, 0, .06998, .8941, 0, 4.8397, .01262   &END
.4675, 0, .06994, .9074, 0, 6.0793, .01449
.1687, 0, .08005, .7203, 0, 1.0635, .00863

10.4   Appendix D: NDARC Model.out File Format

Induced Power Design Variables

    MODEL_ind = 2.0
    Ki_hover = 1.0
    Ki_climb = 1.125
    Ki_prop = 2.0
    Ki_edge = 1.8
    CTs_Hind = 0.05
    kh1 = 1.7
    kh2 = 0.1
    Xh2 = 2.0
    CTs_Pind = 0.149
    kp1 = 1.25
    kp2 = 0.0
    Xp2 = 2.0
    kpa = 0.0
    Xpa = 2.0
    ko1 = 0.0
    ko2 = 8.0
    Maxial = 1.176
    Xaxial = 0.65
    mu_prop = 1.59
    ka1 = 0.0
    ka2 = 0.0
    ka3 = 0.92
    Xa = 5.0
    mu_edge = 0.32
    ke1 = 0.83
    ke2 = 0.0
    ke3 = 1.4
    Xe = 4.8
    kea = 0.0
    Ki_min = 1.085
    Ki_max = 10.0

Profile Power Design Variables

    TECH_drag = 1.0
    Re_ref = 0.0
    MODEL_basic = 2.0
    ncd = 24.0
    CTs_Dmin = 0.026
    d0_hel = 0.008
    d0_prop = 0.003
    d1_hel = 0.0
    d1_prop = 0.1
    d2_hel = 0.31
    d2_prop = 0.25
    dprop = 2.0
    Xprop = 4.0
    CTs_sep = 0.08
    dsep = 1.6
    Xsep = 2.9
    df1 = 0.0
    df2 = 0.0
    Xf = 1.3
    MODEL_stall = 1.0
    nstall = 10.0
    fstall = 1.0
    dstall1 = 2.6
    dstall2 = 60.0
    Xstall1 = 2.2
    Xstall2 = 2.7
    do1 = 0.2
    do2 = 4.7
    dsa = 0.0
    MODEL_comp = 1.0
    fSim = 1.0
    thick_tip = 0.08
    dm1 = 0.01
    dm2 = 0.79
    Xm = 3.1
    Mdd0 = 0.74
    Mddcl = 0.14

10.5 Appendix E: Residuals.out File Format

mux	muz	CT/s	MAT	Offset	Actual Kappa	Actual Cd	Est. Kappa	Kappa Residual	Est. Cd	Cd Residual
0.0	0.0	0.05997	0.6163	0.0	1.0158	0.00837	1.07013	-0.05433	0.00803	0.00034
0.0	0.0	0.06894	0.6163	0.0	1.0549	0.00842	1.08644	-0.03154	0.00808	0.00034
0.0	0.0	0.07814	0.6163	0.0	1.0887	0.0085	1.10256	-0.01386	0.00818	0.00032
0.0	0.0	0.08744	0.6163	0.0	1.1206	0.00867	1.11821	0.00239	0.00832	0.00035
0.0	0.0	0.0969	0.6163	0.0	1.1486	0.00888	1.13343	0.01517	0.00852	0.00036
0.0	0.0	0.10656	0.6163	0.0	1.1724	0.00916	1.14823	0.02417	0.0088	0.00036
0.0	0.0	0.11641	0.6163	0.0	1.1922	0.00963	1.16253	0.02967	0.00916	0.00047
0.0	0.0	0.12644	0.6163	0.0	1.2064	0.01075	1.17624	0.03016	0.00963	0.00112
0.0	0.0	0.13666	0.6163	0.0	1.2172	0.01196	1.18932	0.02788	0.01021	0.00175
0.0	0.0	0.14685	0.6163	0.0	1.228	0.0134	1.20145	0.02655	0.01091	0.00249
0.0	0.0	0.15688	0.6163	0.0	1.2364	0.0152	1.21248	0.02392	0.01224	0.00296
0.0	0.0	0.16665	0.6163	0.0	1.2408	0.01786	1.22234	0.01846	0.01467	0.00319
0.0	0.0	0.1761	0.6163	0.0	1.2431	0.02135	1.23104	0.01206	0.01914	0.00221
0.0	0.0	0.1761	0.6163	0.0	1.2431	0.02135	1.23104	0.01206	0.01914	0.00221
0.0	0.0	0.19258	0.6163	0.0	1.2353	0.035	1.24421	-0.00891	0.035	0.0
0.0	0.0	0.1992	0.6163	0.0	1.2323	0.04284	1.24878	-0.01648	0.0451	-0.00226
0.0	0.0	0.20479	0.6163	0.0	1.2289	0.05138	1.25231	-0.02341	0.05558	-0.0042
0.0	0.0	0.20981	0.6163	0.0	1.2225	0.06283	1.25522	-0.03272	0.06665	-0.00382
0.0	0.0	0.21415	0.6163	0.0	1.2181	0.07734	1.25754	-0.03944	0.07758	-0.00024
0.1687	0.0	0.06998	0.7203	0.0	1.0706	0.00857	1.09663	-0.02603	0.00843	0.00014
0.1927	0.0	0.06997	0.7351	0.0	1.1432	0.00859	1.14613	-0.00293	0.00848	0.00011
0.2167	0.0	0.07002	0.7499	0.0	1.1412	0.00859	1.21278	-0.07158	0.00853	6e-05
0.2407	0.0	0.06994	0.7647	0.0	1.3486	0.00865	1.2997	0.0489	0.00858	7e-05
0.2645	0.0	0.07	0.7794	0.0	1.4106	0.00869	1.41044	0.00016	0.00863	6e-05
0.2883	0.0	0.0699	0.7941	0.0	1.6112	0.00869	1.55058	0.06062	0.00878	-9e-05
0.312	0.0	0.06997	0.8088	0.0	1.8293	0.00884	1.72591	0.10339	0.00898	-0.00014
0.3354	0.0	0.07008	0.8233	0.0	2.0511	0.0091	1.9413	0.1098	0.00921	-0.00011
0.3587	0.0	0.06997	0.8378	0.0	2.3868	0.00936	2.20611	0.18069	0.00952	-0.00016
0.3816	0.0	0.07	0.8522	0.0	2.7063	0.00977	2.52498	0.18132	0.00994	-0.00017
0.4041	0.0	0.07009	0.8664	0.0	3.2166	0.0104	2.90535	0.31125	0.01054	-0.00014
0.4262	0.0	0.07003	0.8804	0.0	3.8686	0.01126	3.35545	0.51315	0.01138	-0.00012
0.4474	0.0	0.06998	0.8941	0.0	4.8397	0.01262	3.87011	0.96959	0.01252	0.0001
0.4675	0.0	0.06994	0.9074	0.0	6.0793	0.01449	4.44454	1.63476	0.01403	0.00046
0.1687	0.0	0.08005	0.7203	0.0	1.0635	0.00863	1.11102	-0.04752	0.00854	9e-05
0.1928	0.0	0.08007	0.7351	0.0	1.1503	0.00866	1.15974	-0.00944	0.00859	7e-05
0.2168	0.0	0.07994	0.7499	0.0	1.2454	0.00867	1.22498	0.02042	0.00863	4e-05
0.2408	0.0	0.07997	0.7647	0.0	1.2312	0.00874	1.31062	-0.07942	0.00868	6e-05
0.2647	0.0	0.07997	0.7795	0.0	1.3395	0.00873	1.42013	-0.08063	0.00874	-1e-05

10.6 Appendix F: Input File with Formatting Errors

```
&RUN_SETTINGS
saveAsFilename = Very Long Run 4    numRuns = 1
   inducedPopulationFactor = 3
   inducedGenerationFactor = 6
   profilePopulationFactor = 6
   profileGenerationFactor = 5
&END

&INDUCED_NDARC_VARIABLES
   MODEL_ind = 2
   Ki_hover = 1.1, 0
   Ki_climb = 1
   Ki_prop = 1
   Ki_edge = 1.6
   CTs_Hind = .12
   kh1 = 0
   kh2 = 1
   Xh2 = 1
   CTs_Pind = .1
   kp1 = 3
   kp2 = 0
   Xp2 = 3.8
   kpa = .62
   Xpa = 1.5
   ko1 = .69
   ko2 = 8.9

   Maxial = 2
   Xaxial = .05
   mu_prop = .1
   ka1 = .42
   ka2 = .76
   ka3 = 0
   Xa = 5.8
   mu_edge = .28
   ke1 = .83
   ke2 = 0
   ke3 = 1.2
   Xe = 3.7
   kea = 0
   Ki_min = .5
   Ki_max = 10
&END

&PROFILE_NDARC_VARIABLES
   TECH_drag = 1
   Re_ref = 0
   MODEL_basic = 2
   ncd = 24
   CTs_Dmin = 0, .1, .01, 5
   d0_hel = 0, .05, .001
   d0_prop = 0, .05, .001
   d1_hel = 3, 2, .01
```

166

```
d1_prop = 0, 2, .01                                  0, 0, .07814, .6163, 0, 1.0887, .0085
d2_hel = .25, .75, .01                               0, 0, .08744, .6163, 0, 1.1206, .00867
d2_prop = .25, .75, .01                              0, 0, .0969, .6163, 0, 1.1486, .00888
dprop = 0, 2, .1                                     0, 0, .10656, .6163, 0, 1.1724, .00916
Xprop = 4, 6, .1                                     0, 0, .11641, .6163, 0, 1.1922, .00963
CTs_sep = 0, .1, .01                                 0, 0, .12644, .6163, 0, 1.2064, .01075
dsep = 1, 4, .1                                      0, 0, .13666, .6163, 0, 1.2172, .01196
Xsep = 2, 5, .1                                      0, 0, .14685, .6163, 0, 1.228, .0134
df1 = 0                                              0, 0, .15688, .6163, 0, 1.2364, .0152
df2 = 0                                              0, 0, .16665, .6163, 0, 1.2408, .01786
Xf = 1.3                                             0, 0, .1761, .6163, 0, 1.2431, .02135
MODEL_stall = 1                                      0, 0, .1761, .6163, 0, 1.2431, .02135
nstall = 10                                          0, 0, .19258, .6163, 0, 1.2353, .035
fstall = 1                                           0, 0, .1992, .6163, 0, 1.2323, .04284
dstall1 = 2.6                                        0, 0, .20479, .6163, 0, 1.2289, .05138
dstall2 = 60                                         0, 0, .20981, .6163, 0, 1.2225, .06283
Xstall1 = 2.2                                        0, 0, .21415, .6163, 0, 1.2181, .07734
Xstall2 = 2.7                                        .1687, 0, .06998, .7203, 0, 1.0706, .00857
do1 = .2                                             .1927, 0, .06997, .7351, 0, 1.1432, .00859
do2 = 4.7                                            .2167, 0, .07002, .7499, 0, 1.1412, .00859
dsa = 0                                              .2407, 0, .06994, .7647, 0, 1.3486, .00865
MODEL_comp = 1                                       .2645, 0, .07, .7794, 0, 1.4106, .00869
fSim = 1                                             .2883, 0, .0699, .7941, 0, 1.6112, .00869
thick_tip = .08                                      .312, 0, .06997, .8088, 0, 1.8293, .00884
dm1 = 0, .25, .01                                    .3354, 0, .07008, .8233, 0, 2.0511, .0091
dm2 = 0, 1, .01                                      .3587, 0, .06997, .8378, 0, 2.3868, .00936
Xm = 2, 4, .1                                        .3816, 0, .07, .8522, 0, 2.7063, .00977
Mdd0 = 0, 2, .01                                     .4041, 0, .07009, .8664, 0, 3.2166, .0104
Mddcl = 0, 2, .01                                    .4262, 0, .07003, .8804, 0, 3.8686, .01126
&END                                                 .4474, 0, .06998, .8941, 0, 4.8397, .01262
                                                     .4675, 0, .06994, .9074, 0, 6.0793, .01449
&CALIBRATION_DATA_SET                                .1687, 0, .08005, .7203, 0, 1.0635, .00863
  #  mu muz CT/s MAT Offset Actual_Kappa Actual_Cd   .1928, 0, .08007, .7351, 0, 1.1503, .00866
  0, 0, .05997, .6163, 0, 1.0158,                    &END
  0, 0, .06894, .6163, 0, 1.0549, .00842
```

10.7   Appendix G: Case Study Calibration Data Set and Design Space

10.7.1   Appendix G.1: Case Study Calibration Data Set

Case	mu	muz	CT/s	MAT	Offset	Actual_Kappa	Actual_Cd
1	0	0	0.05997	0.6163	0	1.0158	0.00837
1	0	0	0.06894	0.6163	0	1.0549	0.00842
1	0	0	0.07814	0.6163	0	1.0887	0.0085
1	0	0	0.08744	0.6163	0	1.1206	0.00867
1	0	0	0.0969	0.6163	0	1.1486	0.00888
1	0	0	0.10656	0.6163	0	1.1724	0.00916
1	0	0	0.11641	0.6163	0	1.1922	0.00963
1	0	0	0.12644	0.6163	0	1.2064	0.01075
1	0	0	0.13666	0.6163	0	1.2172	0.01196
1	0	0	0.14685	0.6163	0	1.228	0.0134
1	0	0	0.15688	0.6163	0	1.2364	0.0152
1	0	0	0.16665	0.6163	0	1.2408	0.01786
1	0	0	0.1761	0.6163	0	1.2431	0.02135
1	0	0	0.1761	0.6163	0	1.2431	0.02135
1	0	0	0.19258	0.6163	0	1.2353	0.035
1	0	0	0.1992	0.6163	0	1.2323	0.04284
1	0	0	0.20479	0.6163	0	1.2289	0.05138
1	0	0	0.20981	0.6163	0	1.2225	0.06283

1	0	0	0.21415	0.6163	0	1.2181	0.07734
2	0.1687	0	0.06998	0.7203	0	1.0706	0.00857
2	0.1927	0	0.06997	0.7351	0	1.1432	0.00859
2	0.2167	0	0.07002	0.7499	0	1.1412	0.00859
2	0.2407	0	0.06994	0.7647	0	1.3486	0.00865
2	0.2645	0	0.07	0.7794	0	1.4106	0.00869
2	0.2883	0	0.0699	0.7941	0	1.6112	0.00869
2	0.312	0	0.06997	0.8088	0	1.8293	0.00884
2	0.3354	0	0.07008	0.8233	0	2.0511	0.0091
2	0.3587	0	0.06997	0.8378	0	2.3868	0.00936
2	0.3816	0	0.07	0.8522	0	2.7063	0.00977
2	0.4041	0	0.07009	0.8664	0	3.2166	0.0104
2	0.4262	0	0.07003	0.8804	0	3.8686	0.01126
2	0.4474	0	0.06998	0.8941	0	4.8397	0.01262
2	0.4675	0	0.06994	0.9074	0	6.0793	0.01449
3	0.1687	0	0.08005	0.7203	0	1.0635	0.00863
3	0.1928	0	0.08007	0.7351	0	1.1503	0.00866
3	0.2168	0	0.07994	0.7499	0	1.2454	0.00867
3	0.2408	0	0.07997	0.7647	0	1.2312	0.00874
3	0.2647	0	0.07997	0.7795	0	1.3395	0.00873
3	0.2886	0	0.08002	0.7942	0	1.4785	0.0088
3	0.3123	0	0.07991	0.8089	0	1.8123	0.00894
3	0.3359	0	0.07997	0.8236	0	2.0282	0.00936
3	0.3594	0	0.08001	0.8382	0	2.2177	0.00948
3	0.3826	0	0.07999	0.8526	0	2.5439	0.00992
3	0.4055	0	0.07985	0.867	0	2.9751	0.01054
3	0.428	0	0.07994	0.8812	0	3.5804	0.01141
3	0.4501	0	0.07989	0.8952	0	4.3215	0.0127
3	0.4713	0	0.07988	0.9089	0	5.3301	0.01456
4	0.1687	0	0.09014	0.7203	0	1.0673	0.00874
4	0.1928	0	0.09	0.7351	0	1.1725	0.00876
4	0.2169	0	0.08999	0.7499	0	1.2369	0.00877
4	0.2409	0	0.08997	0.7648	0	1.3527	0.00886
4	0.2648	0	0.08997	0.7795	0	1.3985	0.0089
4	0.2887	0	0.09003	0.7943	0	1.4718	0.00896
4	0.3126	0	0.08996	0.809	0	1.6821	0.0091
4	0.3363	0	0.08995	0.8237	0	1.9353	0.00952
4	0.3598	0	0.08996	0.8384	0	2.1439	0.00988
4	0.3833	0	0.08997	0.8529	0	2.4108	0.01014
4	0.4065	0	0.08988	0.8674	0	2.8108	0.0108
4	0.4294	0	0.08992	0.8818	0	3.358	0.01173
4	0.4518	0	0.08994	0.8959	0	4.0417	0.01299
4	0.4737	0	0.08992	0.9099	0	4.8654	0.01491
5	0.1687	0	0.1002	0.7203	0	1.0747	0.0089
5	0.1928	0	0.09999	0.7351	0	1.1757	0.00893
5	0.2169	0	0.09999	0.75	0	1.2303	0.00894
5	0.2409	0	0.09998	0.7648	0	1.3401	0.00905
5	0.2649	0	0.09997	0.7796	0	1.4199	0.00914

5	0.2889	0	0.10002	0.7944	0	1.4823	0.00927
5	0.3128	0	0.09996	0.8091	0	1.6677	0.0094
5	0.3365	0	0.09996	0.8239	0	1.9029	0.00969
5	0.3602	0	0.09999	0.8385	0	2.0964	0.01025
5	0.3837	0	0.09999	0.8531	0	2.349	0.01076
5	0.4072	0	0.09991	0.8677	0	2.7359	0.01123
5	0.4303	0	0.09995	0.8822	0	3.2475	0.01233
5	0.4531	0	0.09995	0.8965	0	3.8925	0.01397
5	0.4754	0	0.09992	0.9106	0	4.6806	0.01609
6	0.1688	0	0.1102	0.7203	0	1.0904	0.00917
6	0.1928	0	0.11	0.7351	0	1.1817	0.00918
6	0.2169	0	0.10998	0.75	0	1.2322	0.00921
6	0.241	0	0.11	0.7648	0	1.333	0.00934
6	0.265	0	0.10996	0.7796	0	1.4176	0.00947
6	0.289	0	0.11004	0.7944	0	1.5128	0.00978
6	0.3129	0	0.10999	0.8092	0	1.6806	0.00995
6	0.3367	0	0.10997	0.8239	0	1.8826	0.01031
6	0.3605	0	0.11002	0.8386	0	2.0932	0.01092
6	0.3842	0	0.10999	0.8533	0	2.3445	0.01156
6	0.4077	0	0.10996	0.868	0	2.7063	0.01244
6	0.4311	0	0.10994	0.8825	0	3.2389	0.01413
6	0.4541	0	0.10993	0.8969	0	3.8761	0.01603
6	0.4766	0	0.10996	0.9111	0	4.6874	0.0185
7	0.1688	0	0.12021	0.7203	0	1.1097	0.00962
7	0.1929	0	0.11999	0.7351	0	1.1929	0.00959
7	0.2169	0	0.11999	0.75	0	1.2367	0.00963
7	0.241	0	0.12	0.7648	0	1.3329	0.0098
7	0.265	0	0.11997	0.7796	0	1.4331	0.0103
7	0.289	0	0.11993	0.7945	0	1.6082	0.01071
7	0.313	0	0.11994	0.8093	0	1.7881	0.0112
7	0.3369	0	0.12002	0.824	0	1.8915	0.0122
7	0.3608	0	0.11998	0.8388	0	2.1439	0.01327
7	0.3845	0	0.11999	0.8535	0	2.3895	0.01416
7	0.4081	0	0.11993	0.8681	0	2.7545	0.01529
7	0.4317	0	0.11993	0.8828	0	3.2549	0.0171
7	0.4548	0	0.11993	0.8972	0	3.9212	0.01961
7	0.4776	0	0.11992	0.9115	0	4.6583	0.02248
8	0.1688	0	0.13017	0.7203	0	1.1145	0.01056
8	0.1929	0	0.13002	0.7351	0	1.2071	0.01029
8	0.2169	0	0.13002	0.75	0	1.2414	0.01033
8	0.241	0	0.12995	0.7648	0	1.3532	0.01099
8	0.2651	0	0.12998	0.7797	0	1.4598	0.01194
8	0.2891	0	0.1299	0.7945	0	1.6607	0.01331
8	0.3132	0	0.12999	0.8093	0	1.8249	0.01475
8	0.3371	0	0.12995	0.8241	0	1.9981	0.01617
8	0.3611	0	0.12998	0.8389	0	2.2222	0.01776
8	0.3849	0	0.13001	0.8536	0	2.4644	0.01967
8	0.4086	0	0.12988	0.8683	0	2.8501	0.02107

8	0.4321	0	0.12995	0.883	0	3.3635	0.02403
8	0.4554	0	0.12984	0.8975	0	4.0092	0.02653
8	0.4783	0	0.12987	0.9119	0	4.7688	0.03045
9	0.1688	0	0.14002	0.7203	0	1.173	0.01172
9	0.1929	0	0.14002	0.7352	0	1.2282	0.01176
9	0.217	0	0.14	0.75	0	1.2564	0.01192
9	0.2411	0	0.13995	0.7649	0	1.3917	0.01441
9	0.2651	0	0.13999	0.7797	0	1.5042	0.01635
9	0.2892	0	0.13995	0.7945	0	1.7245	0.01889
9	0.3133	0	0.14002	0.8094	0	1.9076	0.02141
9	0.3373	0	0.13993	0.8242	0	2.104	0.02503
9	0.3613	0	0.13999	0.839	0	2.3129	0.0264
9	0.3852	0	0.13996	0.8538	0	2.5466	0.02951
9	0.409	0	0.13997	0.8685	0	2.9524	0.03444
9	0.4326	0	0.13998	0.8832	0	3.5103	0.03782
9	0.4559	0	0.13994	0.8977	0	4.193	0.04363
9	0.4786	0	0.13999	0.912	0	5.1341	0.05247
10	0.1687	0	0.09012	0.7717	0	1.0729	0.00886
10	0.1928	0	0.08993	0.7876	0	1.1713	0.0089
10	0.2168	0	0.08997	0.8035	0	1.2388	0.00895
10	0.2409	0	0.09	0.8194	0	1.3594	0.00911
10	0.2648	0	0.08999	0.8352	0	1.3376	0.00921
10	0.2887	0	0.08991	0.851	0	1.5016	0.00941
10	0.3125	0	0.08995	0.8668	0	1.7036	0.00982
10	0.3362	0	0.08999	0.8825	0	1.9594	0.01065
10	0.3598	0	0.08997	0.8982	0	2.1831	0.0114
10	0.3831	0	0.08999	0.9137	0	2.4845	0.01266
10	0.4061	0	0.0899	0.9291	0	2.9295	0.01433
10	0.4287	0	0.08995	0.9444	0	3.4817	0.01644
10	0.4508	0	0.08989	0.9595	0	4.242	0.01887
10	0.4722	0	0.08989	0.9742	0	5.1027	0.02188
11	0.1687	0	0.09014	0.7203	0	1.0673	0.00874
11	0.1928	0	0.09	0.7351	0	1.1725	0.00876
11	0.2169	0	0.08999	0.7499	0	1.2369	0.00877
11	0.2409	0	0.08997	0.7648	0	1.3527	0.00886
11	0.2648	0	0.08997	0.7795	0	1.3985	0.0089
11	0.2887	0	0.09003	0.7943	0	1.4718	0.00896
11	0.3126	0	0.08996	0.809	0	1.6821	0.0091
11	0.3363	0	0.08995	0.8237	0	1.9353	0.00952
11	0.3598	0	0.08996	0.8384	0	2.1439	0.00988
11	0.3833	0	0.08997	0.8529	0	2.4108	0.01014
11	0.4065	0	0.08988	0.8674	0	2.8108	0.0108
11	0.4294	0	0.08992	0.8818	0	3.358	0.01173
11	0.4518	0	0.08994	0.8959	0	4.0417	0.01299
11	0.4737	0	0.08992	0.9099	0	4.8654	0.01491
12	0.1687	0	0.08998	0.6661	0	1.093	0.00871
12	0.1687	0	0.09017	0.6661	0	1.061	0.00871
12	0.1928	0	0.08998	0.6799	0	1.1678	0.00873

12	0.2169	0	0.08999	0.6936	0	1.2312	0.00873
12	0.2409	0	0.09	0.7073	0	1.3415	0.00881
12	0.2648	0	0.09	0.7209	0	1.4193	0.00879
12	0.2887	0	0.08997	0.7346	0	1.4713	0.00885
12	0.3126	0	0.08999	0.7482	0	1.6625	0.00893
12	0.3363	0	0.08995	0.7618	0	1.9823	0.00929
12	0.3599	0	0.08998	0.7753	0	2.1418	0.00957
12	0.3834	0	0.08995	0.7888	0	2.3892	0.00963
12	0.4066	0	0.08989	0.8022	0	2.7747	0.00999
12	0.4296	0	0.08991	0.8155	0	3.2782	0.01048
12	0.4522	0	0.08995	0.8287	0	3.9301	0.01106
12	0.4743	0	0.08991	0.8418	0	4.7144	0.01215

10.7.2 Appendix G.2: Case Study Design Space

Rotor Induced Power Variables					
Description	**Variable**	**Lower Bound**	**Upper Bound**	**Resolution**	**Fixed Value**
model (1 constant, 2 standard)	MODEL_ind				2
Induced velocity factors (ratio to momentum theory induced velocity)					
hover	Ki_hover	1	1.3	0.001	
axial climb	Ki_climb				1.08
axial cruise (propeller)	Ki_prop				2
edgewise flight (helicopter)	Ki_edge				2
Variation with Thrust					
CT/s for Ki_h variation	CTs_Hind	0	0.1	0.001	
coefficient for Ki_h	kh1	-8	8	0.001	
coefficient for Ki_h	kh2	-25	25	0.001	
exponent for Ki_h	Xh2	0	4	0.001	
CT/s for Ki_p variation	CTs_Pind	0	0.1	0.001	
coefficient for Ki_p	kp1				0
coefficient for Ki_p	kp2				0
exponent for Ki_p	Xp2				2
Variation with Shaft Angle					
coefficient for Ki_p	kpa				0
exponent for Ki_p	Xpa				2
Variation with Lift Offset					
coefficient for f(offset)	ko1				0
factor for f(offset)	ko2				8
constant in Ki transition from hover to axial cruise	Maxial				1.176
exponent in Ki transition from hover to axial cruise	Xaxial				0.65
Variation with Axial Velocity					
advance ratio for Ki_prop	mu_prop				1
coefficient for Ki(muz) (linear)	ka1				0
coefficient for Ki(muz) (quadratic)	ka2				0
coefficient for Ki(muz)	ka3				0
exponent for Ki(muz)	Xa				4.5
Variation with Edgewise Velocity					
advance ratio for Ki_edge	mu_edge	0	0.45	0.001	
coefficient for Ki(mu) (linear)	ke1	-5	5	0.001	
coefficient for Ki(mu) (quadratic)	ke2	-5	5	0.001	
coefficient for Ki(mu)	ke3	-25	25	0.001	
exponent for Ki(mu)	Xe	4	12	0.001	
variation with rotor drag	kea				0
minimum Ki	Ki_min				1
maximum Ki	Ki_max				10